

## أبعاد الأطوال Length Dimensions

هي كل قيمة رقمية تسند إلى خاصية من خصائص CSS ، على أن تكون ذات طول معين، على أن تتبعها وحدة قياس معينة، ويجب ألا يترك أي فراغ بين القيمة ووحدة القياس، وإلا فإن المتصفح سيتجاهل هذه القيمة، ويسند للخاصية القيمة الافتراضية initial value.

وحدات قياس الأطوال كثيرة، وسوف نعرض معظمها وأكثرها استخداماً :

**1- in :** وحدة تعني inch أو البوصة، والبوصة تساوي 2.54cm.

**2- px :** وحدة تعني pixel والبوصة تساوي 96px.

**3- pt :** وحدة تعني point والبوصة تساوي 72pt.

**4- pc :** وحدة تعني picas والبوصة تساوي 12pc.

**5- em :** وحدة تساوي 16px أي أن القيمة 3em تساوي 48px ، ويمكن أن

تجزية هذه الوحدة كقيمة عشرية، مثل 2.5em ، وهي تعني (16\*2.5) أي تساوي 40px.

ويمكن تغيير القيمة الافتراضية لهذه الوحدة "em" ، عن طريق تغيير قيمة حجم الخط font-size داخل عنصر <body> كالتالي:

```
<body style="font-size: 24px" >
```

وبناء على هذا التغيير تصبح القيمة 3em تساوي 72px.

**6- ch :** وحدة تساوي نصف em عرضاً و 1em طولاً، أي أن قيمة الوحدة

تتغير بتغير قيمة em، وعليه، فإن 1ch يساوي 8px و 2.5ch تساوي 20px.

**7- ex :** هي نفس القيمة السابقة حيث إن 1ex يساوي 0.5em.

**8- rem :** وحدة تساوي القيمة الافتراضية 1em ، أي تساوي 16px ، إلا إذا

قمنا بتغيير القيمة الافتراضية داخل وسم البداية للعنصر <html> كالتالي:

```
<html style="font-size: 10px" >
```

وبناء على هذا التغيير تصبح القيمة 3rem تساوي 30px.

- لاحظ الفرق بين القيمة em و rem ، وهو أن القيمة الافتراضية للوحدة

em يتم تغييرها داخل العنصر <body> ، بينما القيمة الافتراضية للوحدة rem

يتم تغييرها داخل العنصر `<html>` .

**9- vw :** وحدة تعني viewport width وهي تساوي 1% من قيمة عرض المتصفح أي أنه إذا كان عرض المتصفح 800px فهذا يعني أن القيمة 10vw تساوي 1% \* 10 \* 800 أي تساوي 80px ، وهذا يعني أن الخصائص التي يسند إليها قيم ذات وحدة قياس vw ، تتغير قيمتها بتغير عرض المتصفح.

```
<div style = "width: 20vw; background:red">
Change the width of the browser and see the difference!!
</div>
```

قم بتغيير عرض المتصفح ولاحظ تغير عرض العنصر.

**10- vh :** وحدة تعني viewport height ، وهي تساوي 1% من قيمة ارتفاع المتصفح، وهذا يعني أنه إذا كانت قيمة ارتفاع المتصفح 600px فإن القيمة 10vh تساوي 1% \* 10 \* 600 أي تساوي 60px ، وهي تؤثر على عرض وارتفاع الخصائص التي تسند إليها قيم ذات وحدة قياس vh كلما تغير ارتفاع المتصفح.

```
<div style = "width: 20vh; background:red">
Change the height of the browser and see the difference!
</div>
```

**11- % :** وحدة تعني أن قيمة خاصية العنصر الابن، هي نسبة مئوية من قيمة خاصية العنصر الأب الذي يحتويه، أي أنه إذا كان عرض العنصر الأب 500px وكان عرض العنصر الابن 25% فهذا يعني أن عرض العنصر الابن هو 125px، ولذلك فإن أي زيادة أو نقص في قيمة خاصية العنصر الأب، تؤثر على قيمة نفس الخاصية في العنصر الابن، إذا كانت وحدة قياس قيمة الخاصية هي %.

```
<div style = "width: 500px;background: yellow"> This is a parent div
<div style = "width: 25%;background: pink">This is a child div</div>
</div>
```

قم بتغيير عرض العنصر الأب ولاحظ تغير عرض العنصر الابن بنفس النسبة.

**12- vmin :** وحدة تساوي 1% من عرض أو ارتفاع المتصفح أيهم أصغر، أي إذا كان عرض المتصفح 800px وكان ارتفاعه 500px وكان عرض العنصر 50vmin فهذا يعني أن عرض العنصر يساوي (1% \* 500 \* 50) أي أن عرض

العنصر يساوي 250px ، منسوباً للارتفاع لأنه الأصغر قيمة.  
 أي أن عرض العنصر يظل مرتبطاً بتغير قيمة ارتفاع المتصفح زيادة ونقصاناً طالما كانت أقل من عرضه، حتى إذا زادت قيمة الارتفاع عن قيمة العرض، انتقل ارتباط قيمة عرض العنصر من ارتفاع المتصفح إلى عرضه.

```
<div style = "width: 20vmin; background:red">
Change the width and the height of the browser !!
</div>
```

قم بتكبير ارتفاع المتصفح وتصغير عرضه وشاهد تغير عرض العنصر.  
**13 - vmax :** وحدة تساوي 1% من عرض أو ارتفاع المتصفح أيهم أكبر، أي أنه إذا كان عرض المتصفح 800px وكان ارتفاعه 500px وكان عرض العنصر 50vmax فهذا يعني أن عرض العنصر يساوي (1% \* 800 \* 5) أي أن عرض العنصر يساوي 400px منسوباً للعرض لأنه الأكبر قيمة.  
 أي أن عرض العنصر يظل مرتبطاً بتغير قيمة عرض المتصفح زيادة ونقصاناً، طالما كانت أكبر من قيمة ارتفاعه، حتى إذا زادت قيمة الارتفاع عن قيمة العرض، انتقل ارتباط قيمة عرض العنصر، من عرض المتصفح إلى ارتفاعه.

```
<div style = "width: 20vmax; background:red">
Change the width and the height of the browser !!
</div>
```

قم بتكبير ارتفاع المتصفح وتصغير عرضه وشاهد تغير عرض العنصر.

## أبعاد الزوايا Angle Dimensions

يوجد عدة وحدات لقياس زاوية الدوران أو زاوية الميل للعنصر، لاستخدامها مع دوال معينة مثل rotate أو skewX أو أي قيمة يتطلب تحقيقها وجود زاوية معينة مثل قيمة اللون في مزيج الألوان hsl وهذه الوحدات كالتالي:

### -1 degree (deg)

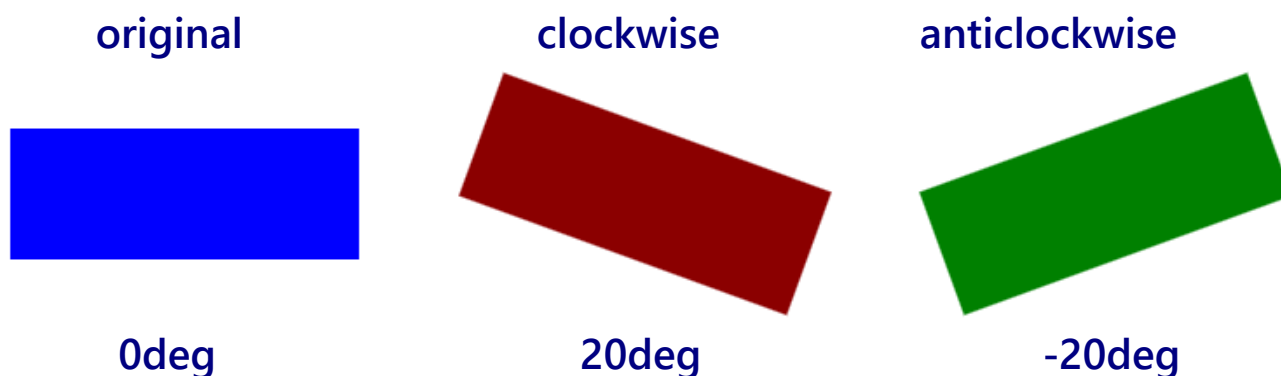
وهي تعني درجة، وتكون قيمتها من 0deg إلى 360deg إذا كان اتجاه دوران الزاوية في اتجاه دوران عقارب الساعة، أو تكون قيمتها من -360deg إلى 0deg إذا كان اتجاه دوران الزاوية عكس اتجاه دوران عقارب الساعة.

#### ملف html

```
<div></div>
<div class="clockw"></div>
<div class="aclockw"></div>
```

#### ملف css

```
div{
  display: inline-block;
  width: 200px;
  height: 75px;
  background-color: blue;
  margin: 30px;
}
.clockw{
  background-color: darkred;
  transform: rotate(30deg);
}
.aclockw{
  background-color: green;
  transform: rotate(-30deg);
}
```



### turn -2

وحدة قياس للزاوية تساوي دورة واحدة كاملة للعنصر أي 360deg، وقيمتها تكون بين 0turn و 1turn ، إذا كان اتجاه دوران الزاوية في اتجاه دوران عقارب الساعة، فنجد أن قيمة الزاوية 0.25turn تساوي 90deg ، وكذلك قيمة الزاوية 0.75turn تساوي 270deg ، وتكون قيمة الوحدة turn بين -1turn و 0turn إذا كان اتجاه دوران الزاوية عكس اتجاه دوران عقارب الساعة، فنجد قيمة الزاوية -0.1turn تساوي -36deg في حين أن قيمة الزاوية -0.5turn تساوي -180turn وعندما نستبدل قيمة الزاوية ووحدة قياسها في الكود السابق (deg) بوحدة القياس الحالية (turn) نحصل على الكود التالي:

```
.clockw{
  background-color: darkred;
  transform: rotate(0.1turn); /* equals 36deg */
}
```

ويكون الكود في حالة اتجاه دوران الزاوية عكس اتجاه دوران عقارب الساعة كالتالي:

```
.aclockw{
  background-color: green;
  transform: rotate(-0.5turn); /* equals -180deg */
}
```

### 3- grad (gradian)

هي وحدة قياس لدوران الزاوية مشابهة لـ deg ، ولكن كل 1grad يساوي 0.9deg ، لذلك تكون قيمة الزاوية محصورة بين 0grad و 400grad إذا كان اتجاه دوران الزاوية في اتجاه دوران عقارب الساعة، وتكون قيمتها بين -400grad و grad إذا كان اتجاه دوران الزاوية عكس اتجاه دوران عقارب الساعة، لذلك نجد أن قيمة الزاوية 100grad تساوي 90deg ، وأيضاً قيمة الزاوية 300grad تساوي 270deg ، في حين أن القيمة -200grad تساوي -180deg .

عندما نستبدل وحدة قياس الزاوية في الكود السابق (turn) بوحدة القياس الحالية (grad) نحصل على الكود التالي:

```
.clockw{  
    background-color: darkred;  
    transform: rotate(200grad);    /* equals 180deg */  
}
```

ويكون الكود في حالة اتجاه دوران الزاوية عكس اتجاه دوران عقارب الساعة كالتالي:

```
.aclockw {  
    background-color: green;  
    transform: rotate(-300grad); /* equals -270deg */  
}
```

#### 4-rad (radians)

وحدة قياس للزاوية تساوي  $\pi / 180$  وحيث إن  $\pi$  تساوي 3.1415 ، فيصبح 1rad يساوي (180/3.1415) أي يساوي (57.296deg) وهو ما يجعل قيمة الوحدة تبدأ من 0rad إلى 6.2832rad تقريباً (أي 360deg) إذا كان اتجاه دوران الزاوية في اتجاه دوران عقارب الساعة، وتكون قيمتها بين -6.2832rad إلى 0rad إذا كان اتجاه دوران الزاوية عكس اتجاه دوران عقارب الساعة، ليكون الكود السابق كالتالي عندما تكون زاوية الدوران تساوي 3.1416 أي 180deg كالتالي:

```
.clockw {  
  width:200px;  
  height:75px;  
  background-color: #ff00ffbb;  
  transform: rotate(3.1416rad); /* equals 180deg */  
}
```

ويكون الكود في حالة اتجاه دوران الزاوية عكس اتجاه دوران عقارب الساعة كالتالي:

```
.aclockw {  
  width:200px;  
  height:75px;  
  background-color: #00ee00bb;  
  transform: rotate(-1rad); /* equals -57.29deg */  
}
```

## أبعاد الوقت Time Dimensions

يوجد عدة أبعاد أو وحدات تقوم بتحديد فترة زمنية معينة، وفي الغالب تستخدم مع خاصيتي animation و transition ، لتحديد الفترة الزمنية لتحريك أو انتقال العناصر ويوجد نوعين منها:

**1- s :** وحدة تعني second أو ثانية، وقد تكون القيمة موجبة (5s) ، وقد تكون سالبة (-2s) ، وقد تتضمن كسراً عشرياً (2.5s).

**2 - ms :** وحدة تعني millisecond ، أو جزء من الألف من الثانية ، وهذه الوحدة قد تكون موجبة (2000ms) أو سالبة (-1500ms) ، ولا داعي لأن نستخدم الكسر العشري مع هذه الوحدة، لأنه يعني 1/1000 من الثانية، وهي فترة زمنية لن يكون الفارق الزمني ملحوظاً بين أجزائها العشرية بالتأكيد.

### ملف html

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

### ملف css

```
div{
  width: 100px;
  height: 100px;
  background: darkred;
  transition: width 2s; /* equals 2000ms */
}
div:hover{
  width: 300px;
}
```



div




div: hover

قم بتحريك المؤشر فوق العنصر div ولاحظ زيادة عرض العنصر من 100px إلى 300px في فترة زمنية قدرها 2 ثانية، ثم قم بزيادة قيمة الثواني إلى 4s ولاحظ



زيادة الفترة التي يزيد فيها عرض العنصر، وإذا استبدلت القيمة 4s بالقيمة 4000ms لن تلاحظ أي تغير في الفترة الزمنية التي تتم فيها زيادة عرض العنصر div عند تحريك المؤشر فوقه.

 اعلم أن أي قيمة مكونة من عدد معين ولا يتبعه وحدة للبعد الزمني لن تعتبر قيمة زمنية وسيتم تجاهلها، وسيتم الاستعاضة عنها بالقيمة الافتراضية للخصائص والتي غالباً ما تكون 0s.

## أبعاد الدقة Resolution Dimensions

يوجد عدة أبعاد تستخدم كقيمة للخصائص resolution أو max-resolution ، التي تستخدم داخل قاعدة @media ، والخاصة بتحديد كيفية ظهور العناصر، عند دقة عرض معينة كالتالي:

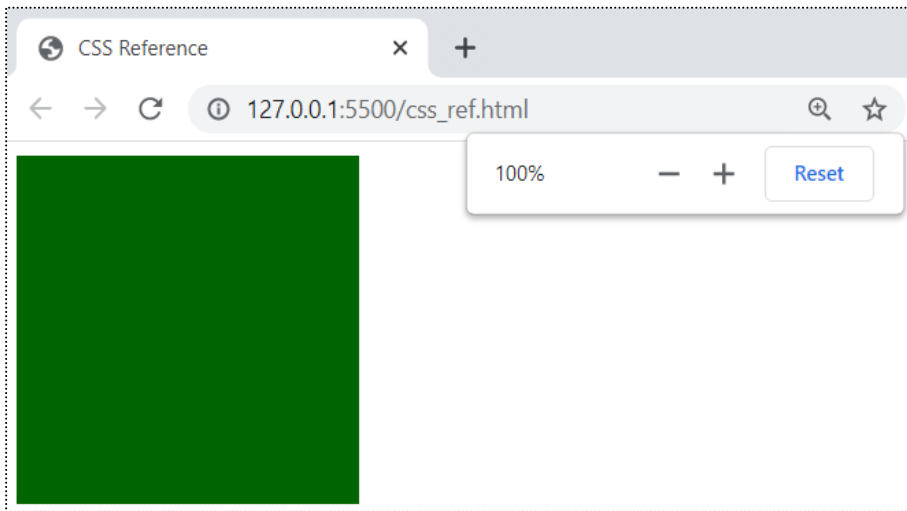
ملف html

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

ملف css

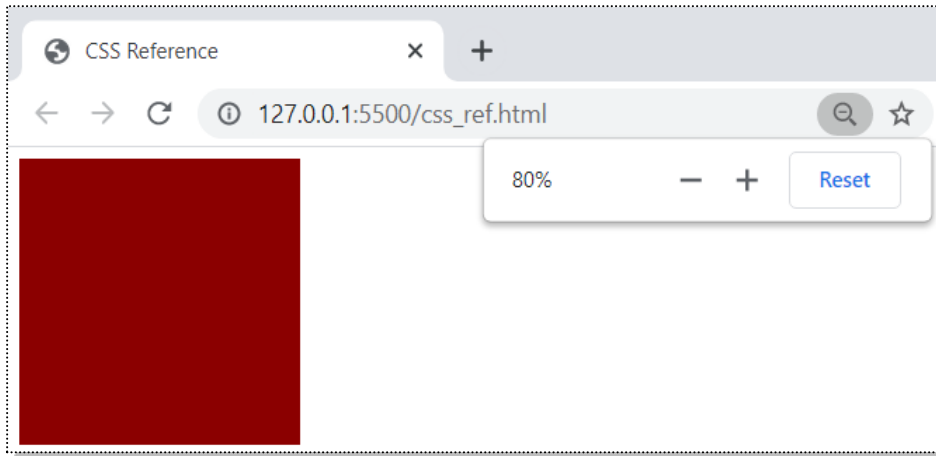
```
div {
    width: 100px;
    height: 100px;
    background: darkgreen;
}

@media(resolution: 96dpi){
    div {
        background: darkred;
    }
}
```



اضغط على زر ctrl مع تحريك عجلة المؤشر حتى تضبط أداة zoom في متصفحك، لتكون 100% ليظهر الشكل بحجمه التقريبي كما بالشكل المقابل.

لاحظ أن لون العنصر هو darkgreen لأن دقة العرض المطلوبة لتغيير اللون هي 96dpi، قم بتصغير وتكبير قيمة zoom ، ولاحظ أنك عندما تصل إلى نسبة 80% سوف يتغير لون العنصر إلى اللون darkred كما بالشكل التالي:



لاحظ أن لون خلفية العنصر تحول إلى اللون darkred عندما وصل resolution أو دقة العرض إلى القيمة 96dpi.

وفي الأغلب يكون هذا هو استخدام أبعاد الدقة resolution وسوف نشرح لاحقاً قاعدة @media ، وباقي قواعد @ في الجزء الثاني من هذا المرجع.

### ملاحظات يجب مراعاتها عند التعامل مع وحدات الأبعاد بأنواعها

أولاً - يجب أن تكون وحدة البعد ملاصقة تماماً للعدد قبلها ولا يفصل بينهما أي مسافة فارغة، وعلى ذلك فالقيم (10 px, 5 s, 45 deg, 2 em, 72 dpi) قيم غير صحيحة وسيتم الاستعاضة عنها بالقيم الافتراضية للخصائص.

ثانياً: يجب أن تكون حروف الوحدة (px, em, deg, grad, ms, dpcm) ملاصقة لبعضها دون أي مسافة فارغة، فالقيم (30px, 45deg, 2.5ms) غير صحيحة وسيتم الاستعاضة عنها بالقيم الافتراضية للخصائص.

ثالثاً: وحدات القياس غير حساسة لحالة الأحرف case-insensitive ، فيمكن تكتب بأحرف كبيرة (9PX, 2EM, 6S) أو صغيرة (5px, 30deg, 25ms) أو مزيج بينهما (15Px, 3eM, 45Deg, 3000Ms, 72Dpi)، ولكن التطبيق والممارسة الأمثل هي استخدام الأحرف الصغيرة دائماً عند كتابة وحدات القياس بشكل عام.

## القيم الثابتة لخصائص CSS

يوجد بعض القيم عبارة عن كلمات محجوزة، لها مدلول معين، فيتم ترجمته لقيمة معينة، وسوف نتناول هذه القيم، وكيفية عملها، ومعرفة الفرق بينها.

### inherit -1

وهي قيمة تسند لخاصية يتم تعيينها للعنصر الابن، وتعني أن الخاصية يسند إليها القيمة المسندة إلى العنصر الأب، الذي يوجد داخله العنصر الابن، والذي تم إسناد نفس الخاصية له.

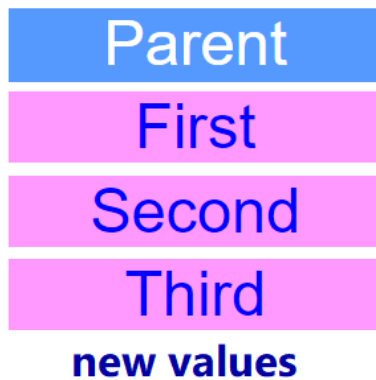
### ملف html

```
<div class= "parent"> Parent
<div class= "first"> First </div>
<div class= "second"> Second </div>
<div class= "third"> Third </div>
</div>
```

### ملف css

```
.parent {
  width: 150px;
  height: 22px;
  color: #fff;
  background-color : #59ff;
  text-align: center;
}
.first, .second, .third{
  width: 150px;
  color: #00f;
  background-color: #f9ff;
  text-align: center;
  margin-top: 5px;
}
```

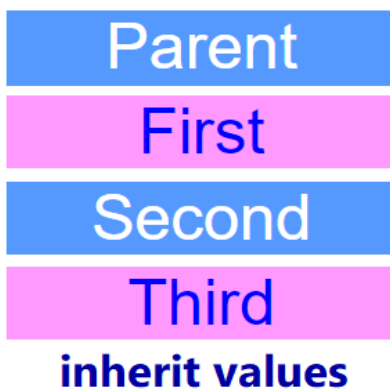
لاحظ أن العنصر الأب Parent يحتوي داخله ثلاثة أبناء، وقد تم تنسيق العناصر الأبناء بتنسيق مختلف عن العنصر الأب، ولكنه نفس التنسيق للعناصر



الأبناء جميعاً، نفس لون النص للخاصية color ونفس لون الخلفية للخاصية background-color .

قم بإسناد القيمة inherit للخاصية color والخاصية background-color للعنصر Second فقط، ولاحظ الفرق في تنسيق العنصر:

```
.second{
  color: inherit;
  background-color: inherit;
}
```



لاحظ أنه بعد إسناد القيمة inherit إلى خاصيتي لون النص color ولون الخلفية background-color للعنصر Second ، أصبح للعنصر نفس لون النص، ونفس لون الخلفية للعنصر الأب Parent . مما يعني أن القيمة inherit تهتم أولاً وأخيراً بقيمة العنصر الأب، وتقوم بإسنادها للعنصر الابن،

وإن لم يكن العنصر المستهدف ينحدر من عنصر أب، تقوم القيمة inherit باستخدام القيم المخصصة للمستخدم، ثم القيم الافتراضية للمتصفح.

### الفرق بين القيمة inherit والخاصية الموروثة inherited

**inherit** : هي قيمة يتم إسنادها إلى أي خاصية غير موروثة، يتم تعيينها داخل العنصر الابن، لنقل قيمة نفس الخاصية من العنصر الأب إلى العنصر الابن.

**inherited** : هي صفة لبعض الخصائص تعني أن الخاصية موروثة، أي أن قيمتها للعنصر الأب، سيتم توريثها إلى العناصر الأبناء المنحدرة منه تلقائياً، دون أن يتم تعيينها داخلهم، مثل الخاصية color والخاصية font-size .

## initial -2

إذا كان قد تم تعيين خاصية معينة لعنصر ما، وتم إسناد قيمة معينة لها، فيمكن إعادة قيمة الخاصية إلى القيمة المخصصة للمستخدم، بإسناد القيمة initial لها، فإذا لم يسند المستخدم للخاصية قيم معينة، فيتم إعادة قيمتها إلى القيمة الافتراضية للمتصفح، والقيمة initial تتجاهل تنسيقات العنصر الأصلي وتنسيقات العنصر الأب.

### ملف html

```
<div>
  <kbd> normal</kbd>
  <kbd class="initial"> initial </kbd>
</div>
```

### ملف css

```
div {
  color: green;
  font-family : times;
}
kbd {
  display : block;
  width: 120px;
  margin: 10px;
  text-align: center;
  background-color: pink;
  color : blue;
  font-family : impact;
}
.initial{
  background-color: initial;
  color : initial;
  font-family : initial;
}
```



بالطبع أنت تعلم أن نوع  
الخط الافتراضي للعنصر  
<kbd> هو monospace  
ولكنني قمت بتغيير نوع

الخط من إعدادات المتصفح ليكون ink free كما بالصورة، وقمت بتغيير نوع  
الخط للعنصر الأصلي إلى impact وأسندت للعنصر <div> الأب الذي يحتوي  
العنصرين <kdb> نوع خط times.

**normal**

initial

كما قمت بمنح العنصر الأصلي لون نص  
blue ولون خلفية pink ومنحت العنصر  
<div> الأب لون نص green، وتركت العنصر  
الأول normal دون أي تغيير في قيم

الخصائص، وأسندت للعنصر الثاني initial القيمة initial لكل الخصائص  
السابقة، ليظهر العنصران كما بالشكل المقابل.

ليتضح من هذا الشكل، أن القيمة initial تجاهلت كل القيم التي تم إسنادها  
للخاصية font-family ، كما بالكود السابق، واستخدمت القيمة المخصصة  
للمستخدم والتي قمت بتغييرها في إعدادات المتصفح وهي ink free .  
وبالنسبة للخاصية color تجاهلت القيمة initial كل القيم واستخدمت  
القيمة black وهي القيمة الافتراضية للمتصفح، لأنه لا توجد قيمة مخصصة  
للمستخدم للخاصية color.

وبالنسبة للخاصية background-color تجاهلت القيمة initial كل القيم  
واستخدمت القيمة transparent ، وهي القيمة الافتراضية للمتصفح، لأنه لا  
توجد قيمة مخصصة للمستخدم للخاصية background-color.

ومما سبق يتضح أن القيمة initial تهتم أولاً بالقيمة المخصصة للمستخدم،  
يليها القيمة الافتراضية للمتصفح، وتتجاهل قيم العنصر الأصلي والعنصر الأب.

**unset -3**

إذا تم تعيين خاصية موروثة inherited للعنصر الابن، فإن إسناد القيمة unset لها، تسند لهذه الخاصية قيمة نفس الخاصية في العنصر الأب، الذي يحتوي هذا العنصر الابن، وإذا لم يتم تعيين الخاصية في العنصر الأب، تعيد القيمة unset قيمة الخاصية إلى القيم المخصصة للمستخدم، فإذا لم يكن للمستخدم قيم مخصصة لهذه الخاصية، تعيد unset قيمة الخاصية إلى القيم الافتراضية للمتصفح.

أما إذا كانت الخاصية غير موروثة non-inherited فإن القيمة unset تعيد قيمة الخاصية إلى قيم المتصفح الافتراضية.

باستخدام الكود السابق، استبدل القيمة initial للخاصية class بالقيمة unset ، واستبدلها في كود CSS، واترك باقي الكود دون تغيير كالتالي:

**ملف html**

```
<div>
<kbd> normal</kbd>
<kbd class="unset"> unset</kbd>
</div>
```

```
div {
    color: green;
    font-family : times;
}
kbd {
    display : block;
    width: 120px;
    margin: 10px;
    text-align: center;
    background-color: pink;
    color : blue;
    font-family : impact;
}
.unset{
```



```
background-color: unset;
color : unset;
font-family : unset;
}
```

## normal

### unset

يتضح من هذا الشكل، أن القيمة `unset` تجاهلت كل القيم التي تم إسنادها للخاصية `font-family` ، كما بالكود السابق، واستخدمت قيمة العنصر الأب، وهي `times` .

وبالنسبة للخاصية `color` تجاهلت القيمة `unset` كل القيم واستخدمت القيمة `green` وهي أيضاً قيمة العنصر الأب.

وبالنسبة للخاصية `background-color` تجاهلت القيمة `unset` كل القيم واستخدمت القيمة `transparent` ، وهي القيمة الافتراضية للمتصفح.

ومما سبق يتضح أن القيمة `unset` للخاصية الموروثة تهتم أولاً بقيمة الخاصية للعنصر الأب، ثم القيمة المخصصة للمستخدم، ثم القيمة الافتراضية للمتصفح، أما الخاصية غير الموروثة، فالقيمة `unset` تذهب مباشرة إلى القيمة الافتراضية للمتصفح.

## revert -4

إذا تم تعيين خاصية موروثة inherited للعنصر الابن، فإن إسناد القيمة revert لها، يسند لها القيمة الافتراضية للعنصر نفسه، فإذا لم يكن للعنصر قيمة افتراضية لهذه الخاصية، تسند لها قيمة الخاصية في العنصر الأب، وإن لم يتم تعيينها للعنصر الأب، تسند لها القيمة الافتراضية للمتصفح.

أما إذا كانت الخاصية غير موروثة non-inherited فإن القيمة revert تسند للخاصية قيمة المتصفح الافتراضية.

باستخدام الكود السابق، استبدل القيمة unset للخاصية class بالقيمة revert ، واستبدلها في كود CSS، واترك باقي الكود دون تغيير كالتالي:

## ملف html

```
<div>
<kbd> normal</kbd>
<kbd class="revert"> revert </kbd></div>
```

## ملف css

```
div {
    color: green;
    font-family : times;
}
kbd {
    display : block;
    width: 120px;
    margin: 10px;
    text-align: center;
    background-color: pink;
    color : blue;
    font-family : impact;
}
.revert{
    background-color: revert;
    color : revert;
    font-family : revert;
}
```

**normal****revert**

يتضح من هذا الشكل، أن القيمة `revert` تجاهلت كل القيم التي تم إسنادها للخاصية `font-family` ، كما بالكود السابق، واستخدمت القيمة الافتراضية للعنصر وهي `monospace` .

وبالنسبة للخاصية `color` تجاهلت القيمة `revert` كل القيم واستخدمت القيمة `green` وهي قيمة العنصر الأب.

وبالنسبة للخاصية `background-color` تجاهلت القيمة `revert` كل القيم واستخدمت القيمة `transparent` ، وهي القيمة الافتراضية للمتصفح.

ومما سبق يتضح أن القيمة `revert` للخاصية الموروثة تهتم أولاً بالقيمة الافتراضية للعنصر نفسه، ثم قيمة الخاصية للعنصر الأب، ثم القيمة المخصصة للمستخدم، ثم القيمة الافتراضية للمتصفح، أما الخاصية غير الموروثة، فالقيمة `revert` تذهب مباشرة إلى القيمة الافتراضية للمتصفح.

### القيمة الافتراضية للعنصر والقيمة الافتراضية للمتصفح:

بالرغم من أن تقسيم أصول التنسيق في MDN، لم يشر إلى القيم الافتراضية لخصائص العنصر، إلا أنه بالتجربة وجدت أن إسناد القيمة `revert` إلى العنصر أعادت الخاصية `font-family` إلى القيمة `monospace` وهي القيم الافتراضية للخاصية `font-family` في العنصر `<pre>` ، بينما أعادت الخاصية `color` إلى القيمة `green` وهي قيمة الخاصية للعنصر الأب، ولم يعدها للقيمة الافتراضية للمتصفح وهي اللون الأسود، بالرغم من أن كلاهما خاصية موروثة، فظهر واضحاً الفرق بين القيمة الافتراضية للعنصر والقيمة الافتراضية للمتصفح.

## currentColor -5

قيمة تعني أن الخاصية يسند إليها قيمة اللون المسندة للخاصية color ، وهي قيمة لون النص بين وسمي البداية والنهاية للعنصر، فعند إسناد هذه الخاصية مثلاً للون الحدود أو لون الظل، نجد أنه نفس لون النص في العنصر، وكذلك عند إسنادها لأي من خصائص الألوان ما عدا الخاصية background-color ، لأنها سوف تخفي النص الموجود داخل العنصر، لأنه سيكون بنفس لون الخلفية.

ملف html

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

ملف css

```
div {  
  width : 200px;  
  height :50px;  
  color : #808;  
  border: 10px dashed currentColor;  
  background-color : #59f8;  
  text-align: center;  
  padding-top: 20px;  
  margin: 20px;  
}
```



لاحظ أن القيمة currentColor المسندة لخاصية لون الحدود border ، جعلت لون الحدود هو نفسه لون النص color ، وعموماً فالقيمة currentColor هي القيمة الافتراضية للون الحدود، أي أنه حتى في حالة عدم كتابتها في الكود فإن الحدود ستتلون بنفس لون الخاصية color.

## الخاصية all

خاصية تمثل كل خصائص CSS ماعدا المتغيرات CSS variables والخاصية unicode-bidi، حيث إن إسناد أي قيمة للخاصية all ، يعني إسناد هذه القيمة لجميع خصائص CSS .

## ملف html

```
<div>
<p class= "first"> First </p>
<p class= "second"> Second </p>
</div>
```

## ملف css

```
div{
    width : 185px;
    color : blue;
    background-color :pink;
    margin: 5px;
    font-family: arial;
    border: 2px solid currentColor;
}
p{
    color : darkred;
    font-family: arial;
    background-color : gold;
    margin: 5px;
    text-align: center;
    border: 2px solid blue;
}
.first, .second{
    color : lightgreen;
    background-color : olive;
    font-family: impact;
    border: 2px solid currentColor;
}
```

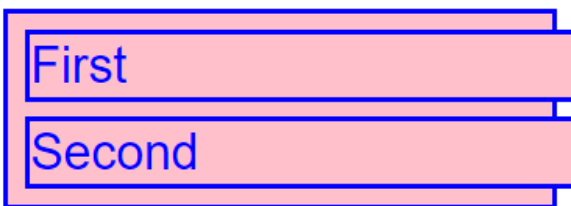
لاحظ أننا أنشأنا عنصر <div> أب يحتوي بداخله عنصرين <p> وللعنصر



الأب تنسيقاته الخاصة، وللعنصرين الأبناء تنسيقاتهما الخاصة.

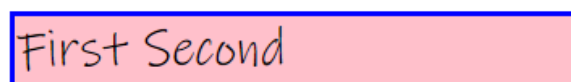
قم بتعيين الخاصية `all` للعنصرين الأبناء واسند لها القيم `initial` و `inherit` و `unset` و `revert` ولاحظ الفرق في تأثير وسلوك كل قيمة على حدة .

`all: inherit;`



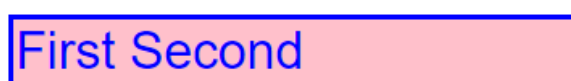
لاحظ أن تعيين القيمة `inherit` للخاصية `all` أسند للعناصر الأبناء كل قيم العنصر الأب لكل الخصائص.

`all: initial;`



لاحظ أن تعيين القيمة `initial` للخاصية `all` أسند للعناصر الأبناء كل القيم المخصصة للمستخدم، وفي حالة عدم وجودها تم إسناد القيم الافتراضية للمتصفح لباقي الخصائص.

`all: unset;`



لاحظ أن تعيين القيمة `unset` للخاصية `all` أسند للخصائص الموروثة في

العناصر الأبناء قيمة نفس الخاصية في العنصر الأب، مثل `color` و `font-family` أما الخصائص الغير موروثة فقد أسند لها القيم المخصصة للمستخدم، وفي حالة عدم وجودها تم إسناد القيم الافتراضية للمتصفح لها، مثل الخاصية `background-color` ، والخاصية `border` ، والخاصية `margin` .

`all: revert;`



لاحظ أن تعيين القيمة `revert` للخاصية `all` أسند للخصائص في العناصر الأبناء الخصائص الافتراضية للعنصر نفسه سواء كانت الخصائص

موروثة أو غير موروثة، فأعاد العناصر الأبناء إلى الهامش العلوي والسفلي الافتراضيين للعنصر `<p>` ، وفي حالة عدم وجود قيمة افتراضية للعنصر لأي خاصية موروثة، يتم إسناد قيم العنصر الأب لهذه الخاصية، مثل الخاصية `color` ، وفي حالة عدم وجود قيم للخاصية في العنصر الأب، يتم إسناد القيم المخصصة للمستخدم، ثم القيم الافتراضية للمتصفح.

لاحظ أن القيمة `revert` قد فرقت بين القيمة الافتراضية للعنصر والقيمة الافتراضية للمتصفح، ففي حالة العناصر التي لها قيمة افتراضية لخاصية معينة، فإن القيمة `revert` تذهب إليها مباشرة، وفي حالة عدم وجود هذه القيمة تذهب مباشرة إلى قيمة العنصر الأب، ثم القيم المخصصة للمستخدم، ثم القيم الافتراضية للمتصفح في حالة الخصائص الموروثة، أم الخصائص الغير موروثة، تتبع نفس المسار ولكنها لا تمر على قيم الخاصية في العنصر الأب.

**writing-mode**

خاصية تحدد اتجاه النص داخل العنصر، سواء كان أفقياً أو رأسياً والخاصية لها قيمة افتراضية horizontal-tb وتقبل عدة أنواع من القيم كالتالي:

- **horizontal-tb** : قيمة تعني أن النص يكون أفقياً داخل العنصر.

- **vertical-rl** : قيمة تعني أن النص يكون رأسياً واتجاهه من اليمين لليسار.

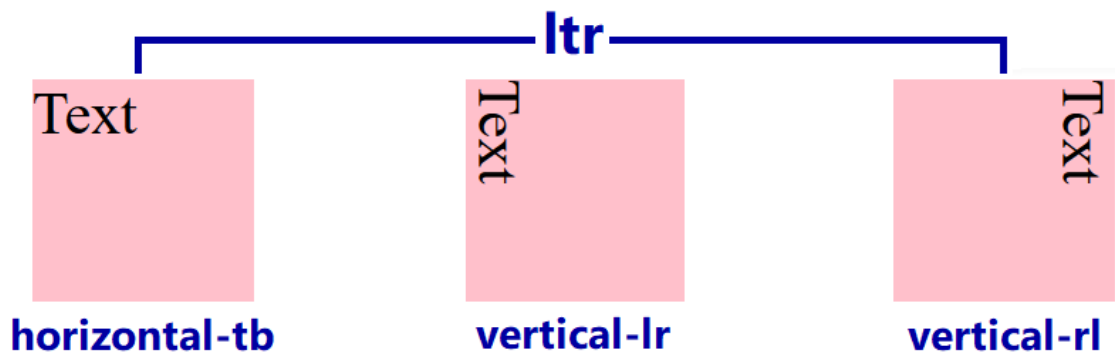
- **vertical-lr** : قيمة تعني أن النص يكون رأسياً واتجاهه من اليسار إلى اليمين.

ملف html

```
<div> Writing Mode </div>
```

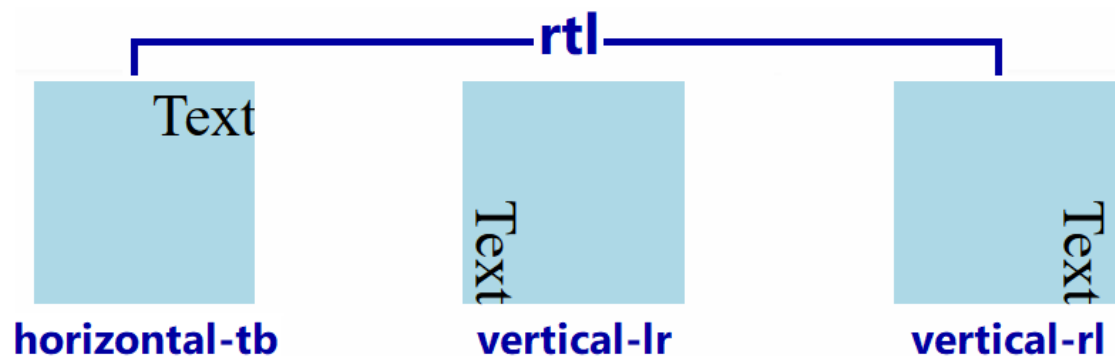
ملف css

```
div{
  width: 60px;
  height: 60px;
  background-color: pink;
  direction: ltr;
  writing-mode: horizontal-tb;
}
```



اتجاه محتوى العنصر direction يؤثر على مكان النص سواء كان النص أفقياً أم

رأسياً

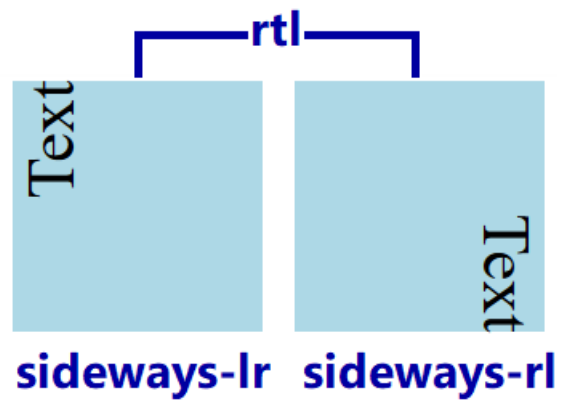
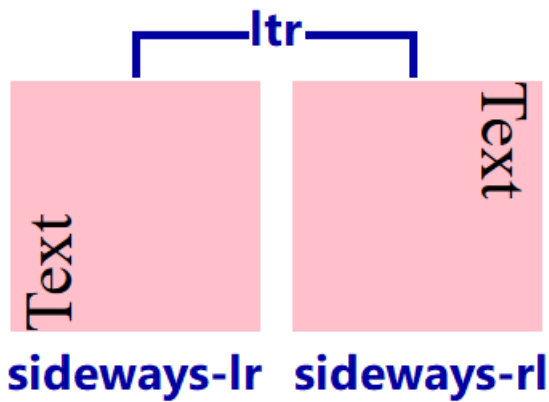




يوجد لهذه الخاصية قيمتين أخريين هما `sideways lr` و `sideways rl` ولكنهما مدعومتان في متصفح فايرفوكس فقط، وتأثيرهما كالتالي:

- ***sideways rl*** : إذا كان اتجاه النص من اليمين لليسار، فيبدأ النص من الأسفل للأعلى، وإذا كان اتجاه النص من اليسار إلى اليمين، فيبدأ النص من الأعلى إلى الأسفل.

- ***sideways lr*** : إذا كان اتجاه النص من اليمين إلى اليسار فيبدأ النص من الأعلى للأسفل، وإذا كان اتجاه النص من اليسار إلى اليمين فيبدأ النص من الأسفل إلى الأعلى.



## border-image

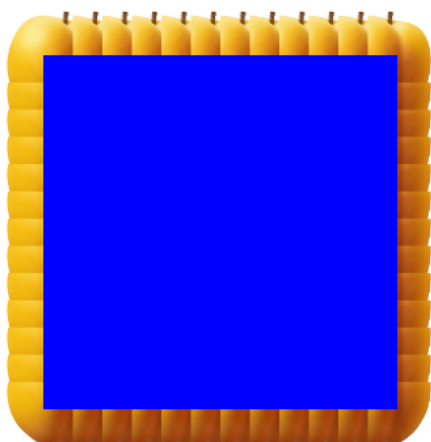
خاصية تقوم بتحديد العنصر بصورة image بدلاً من الحدود الثابتة التقليدية والتي تعرفنا عليها من قبل:

ملف html

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

ملف css

```
div{  
  color: #fff;  
  width: 50px;  
  height: 50px;  
  background-color: blue;  
  margin: 20px;  
  border: 20px groove orange;  
  border-image: url(apricot.png) 80/ 15px / 13px round;  
}
```



وهذه الخاصية تضم عدة خصائص تعمل على إظهار الحدود بالشكل المطلوب:

```
div{  
  color: #fff;  
  width: 50px;  
  height: 50px;  
  background-color: blue;  
  margin: 20px;  
  border: 20px groove orange;  
  border-image: url("heart.png");  
  border-image-slice: 140;
```

```
border-image-width: 7px;  
border-image-outset: 5px;  
border-image-repeat: round;  
}
```

وهذه الخصائص كالتالي:

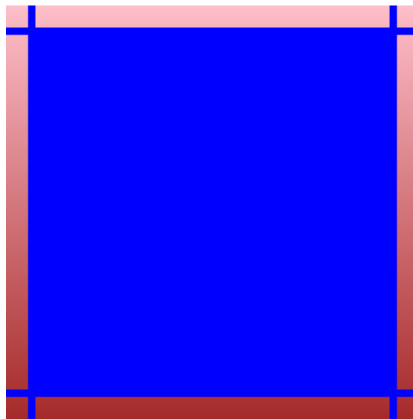
## border-image-source

خاصية تحدد مصدر الصورة المستخدمة في عمل الحدود، ونسند لها قيمة تمثل عنوان أو المسار إلى هذه الصورة بين قوسين قبلهما url:

```
border-image-source: url("images/border.png");
```

والمصدر قد يكون تدرج لوني linearGradient:

```
border-image-source: linear-gradient(pink, brown);
```



## border-image-slice

خاصية تحدد عرض الجزء الذي سيتم قصه من الصورة على كل جوانبها ليملاً حد العنصر في نفس الجانب منه ونسند لها قيمة مكونة من رقم إلى أربع أرقام، كل رقم يمثل عرض الصورة على كل اتجاه بالترتيب:

top – right – bottom – left

```
border-image-slice: 10 15 20 10;
```

هذه الأرقام للتوضيح ودائماً تتوقف على أبعاد الصورة، فالأرقام السابقة تعني أننا سنقوم بقص 10px من أعلى الصورة لنملاً به الحد

العلوي للعنصر، كما أننا سنقوم بقص 15px من الجانب الأيمن للصورة لنملاً به الجانب الأيمن للعنصر، كما أننا سنقوم بقص 20px من أسفل الصورة لنملاً بها الحد السفلي للعنصر، كما أننا سنقوم بقص 10px من الجانب الأيسر للصورة لنملاً به الجانب الأيسر للعنصر.

قد نكتفي بثلاث قيم ونستغني عن القيمة الرابعة في حالة تساوي عرض الجزء المقصوص في الاتجاه right مع عرض الجزء المقصوص في الاتجاه left.

```
border-image-slice: 15px 17px 10px;
```

قد نكتفي بقيمتين اثنتين عند تساوي عرض الجزء المقصوص في الاتجاه top مع عرض الجزء المقصوص في الاتجاه bottom وتمثلهم القيمة الأولى، وعند تساوي عرض الجزء المقصوص في الاتجاه right مع مثيله في الاتجاه left وتمثلهم القيمة الثانية.

```
border-image-slice: 15px 10px;
```

قد نكتفي بقيمة واحدة فقط عند تساوي عرض الأجزاء المقصوصة في كل الاتجاهات.

```
border-image-slice: 15px;
```

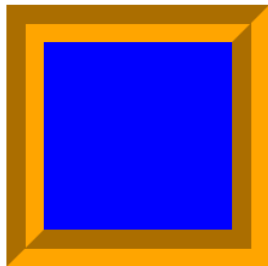
يمكن التعبير عن قيمة الجزء المقصوص باستخدام كنسبة المئوية من حدود الصورة.

```
border-image-slice: 20% 15% 35% 10%;
```

يمكن إضافة الكلمة المفتاحية fill ليتم ملء خلفية العنصر بالجزء الذي لم يتم قصه من الصورة أو تكراراته حسب عرض الأجزاء المقصوصة من كل جانب.



image



element



slice no-fill



slice fill

## border-image-width

خاصية تحدد عرض حد العنصر الذي سيتم ملؤه بالجزء المقصوص من الصورة.

```
border-image-width: 15px 10px 20px 15px;
```

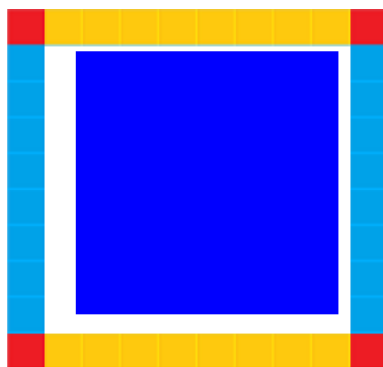
وهذه الخاصية مثل الخاصية border-image-slice يمكن الاكتفاء بثلاث قيم أو اثنتين أو واحدة، طبقاً لنفس الشروط السابق شرحها.

## border-image-outset

خاصية تحدد عرض المساحة الفارغة بين الحدود وخلفية العنصر.

```
border-image-outset: 15px 10px 20px 15px;
```

ومثلها مثل سابقتها في تنوع عدد القيم المسندة إليها وب نفس الشروط.

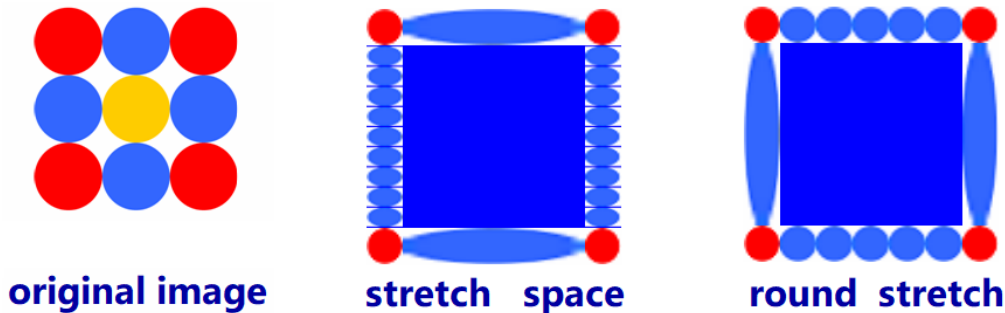


## border-image-repeat

خاصية تحدد كيف سيتم ملء الحدود بالأجزاء المقصوصة من الصورة عندما يكون عرض الجزء المقصوص image-border-slice أصغر

## من عرض حدود العنصر image-border-width

```
border-image-outset: stretch space;
```



وهذه الخاصية لها عدة قيم:

- **stretch**: هي القيمة الافتراضية، وتعمل هذه القيمة على استطالة

الجزء المقصوص بحيث يملأ عرض الحدود.

- **repeat**: تعمل هذه القيمة على تكرار الجزء المقصوص داخل

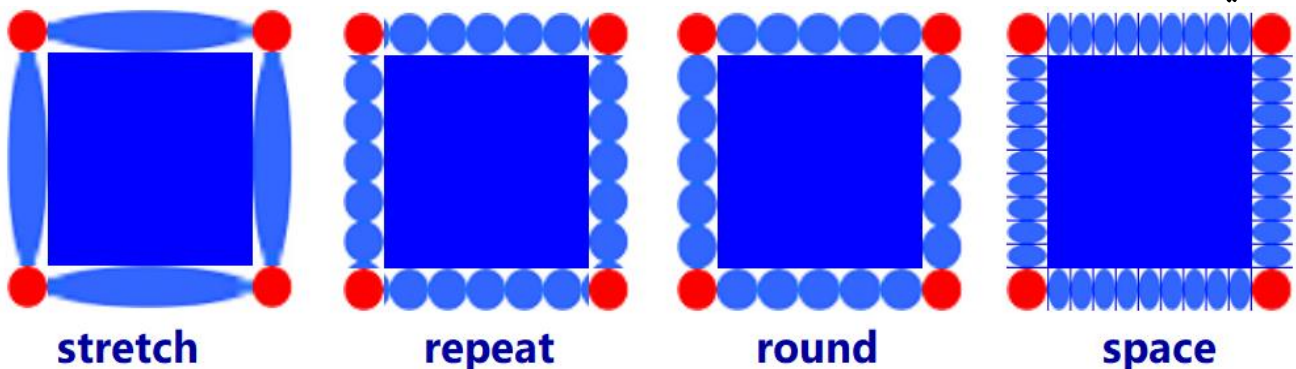
الحدود حتى يتم ملء عرض الحدود كوحدة كاملة متكررة من الجزء المقصوص أو أجزاء من الوحدة لاستكمال ملء عرض الحدود.

- **round**: تعمل هذه القيمة على تكرار الجزء المقصوص داخل الحدود

حتى يتم ملء عرض الحدود كوحدة كاملة وتقوم باستطالة الوحدات الكاملة لملء أي جزء صغير من الحدود لا يستوعب وحدة كاملة من الجزء المقصوص.

- **space**: تعمل هذه القيمة على تكرار الجزء المقصوص داخل الحدود

حتى يتم ملء عرض الحدود كوحدة كاملة مع ترك فراغ بين كل وحدة والتي تليها من الجزء المقصوص.



وهذه الخاصية لا تقبل إلا قيمة واحدة لملء كل الحدود، أو قيمتين، حيث تمثل القيمة الأولى الحد العلوي والحد السفلي، والقيمة الثانية تمثل الحد الأيمن والحد الأيسر.

ولا تقبل هذه الخاصية ثلاث قيم أو أربعة بخلاف الخصائص السابقة.

## التدرج اللوني المخروطي conic-gradient

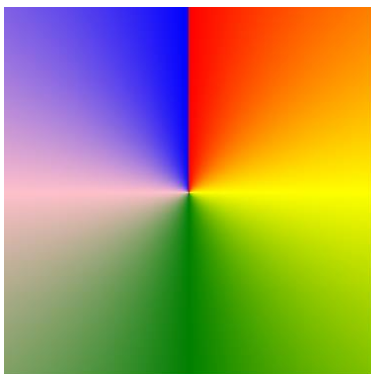
هو تدرج لوني على شكل مخروط مركزه افتراضياً هو مركز العنصر وخط بدايته افتراضياً هو نصف قطر العنصر الرأسى وهو نفسه خط نهاية التدرج افتراضياً، وتتوزع الألوان على محيط دائرة التدرج على شكل مثلث متساوي الضلعين.

ملف html

```
<div> linear gradient </div>
```

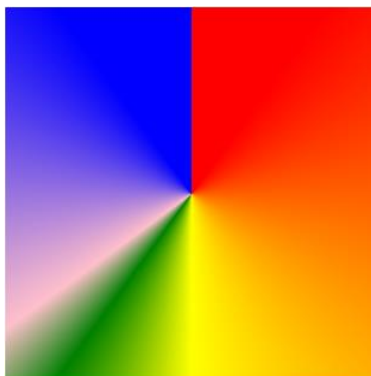
ملف css

```
div {
  width: 200px;
  height: 200px;
  background-image: conic-gradient(red, yellow, green,
    pink , blue);
}
```



ويتم تقسيم الألوان في التدرج بنسبة قوس مخروط اللون إلى محيط دائرة التدرج كالتالي:

```
background-image: conic-gradient(red 10%, yellow 50%,
  green 60%, pink 65%, blue 90%);
```



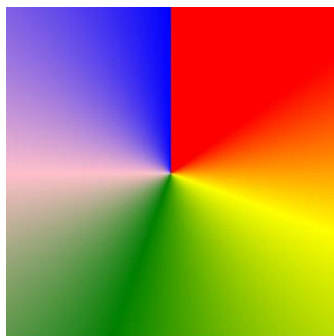
لاحظ أن هذه النسب ما هي إلا نسبة إزاحة خط نشأة اللون (نصف قطر اللون) عن خط نشأة التدرج إلى محيط دائرة التدرج (خط نشأة التدرج عبارة عن نصف قطر بدايته مركز التدرج ونهايته أعلى نقطة في محيط دائرة التدرج في منتصف الحد العلوي للعنصر).

وبداية نسبة التدرج افتراضياً هي 0% ونهايتها الافتراضية هي 100% أو يمكن تقسيم الألوان عن طريق تعيين الزاوية المحصورة بين خط بداية



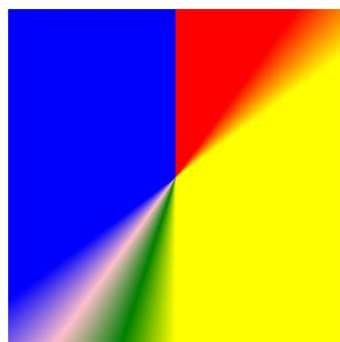
التدرج وخط نشأة اللون وقيمة الزاوية تكون من 0deg إلى 360deg.

```
background-image: conic-gradient(red 55deg, yellow 110deg,
green 200deg, pink 270deg, blue);
```



يمكن استخدام أي وحدة قياس للزاوية في تحديد زاوية اللون في التدرج. ويمكن تحديد مساحة معينة للون بتحديد نقطتي نشأة ضلعي مثلث اللون كالتالي:

```
background-image: conic-gradient(red 10%, yellow 15% 50%,
green 55%, pink 60%, blue 65% 90%);
```

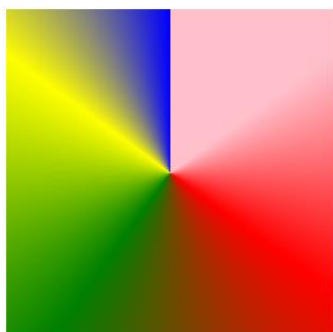


لاحظ أن خط بداية اللون الأصفر يقع على مسافة من خط نشأة التدرج تساوي 15% من محيط التدرج، بينما إزاحة خط نهايته عن خط نشأة التدرج تساوي 50% من محيط التدرج، كما أن نقطة بداية اللون الأزرق تقع على مسافة تساوي 65% من محيط التدرج بينما نقطة خط النهاية تقع على مسافة تساوي 90% من محيط التدرج.

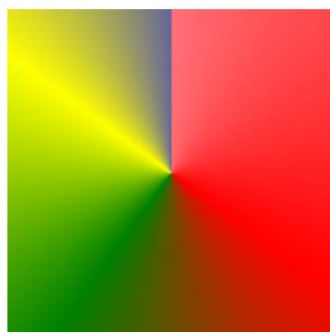
### تحديد نسبة أو زاوية بداية ونهاية التدرج

ذكرنا سابقاً أن نسبة بداية خط نشأة التدرج هي 0% افتراضياً، وأن نسبة نهايته هي 100% افتراضياً، ولكن يمكن أن يبدأ التدرج بنسبة أقل من 0% ويمكن أن ينتهي أكبر من 100% كالتالي:

```
background-image: conic-gradient(pink -25%, red 35%,
green 60%, yellow 85%, blue 110%);
```



0% : 100%

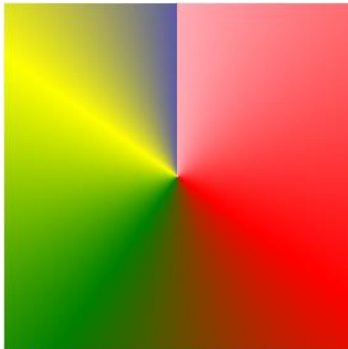


-25% : 110%

لاحظ في الشكل الثاني أن اللون pink في بداية التدرج أصبح أقل قوة لأن خط نشأة اللون الأول في التدرج أصبح قبل بداية التدرج بنسبة تساوي 25% من محيط التدرج، كما أن اللون blue في نهاية التدرج أصبح أقل قوة لأن خط نشأة اللون في نهاية التدرج أصبح بعد نهاية التدرج بنسبة تساوي 10% من محيط التدرج.

لاحظ أنه يمكنك أن نستخدم زاوية اللون بنفس الطريقة لنبدأ التدرج قبل 0deg وننتهي بعد 360deg كالتالي:

```
background-image: conic-gradient(pink -15deg, red 125deg,
green 215deg, yellow 305deg, blue 390deg);
```



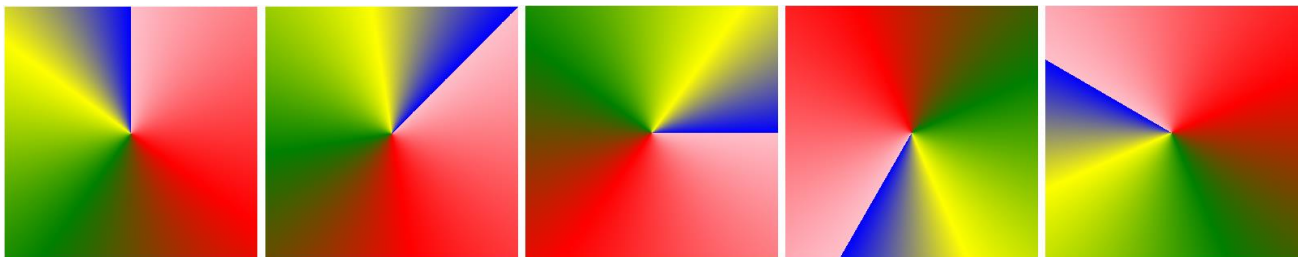
-15deg : 390deg

وبتطبيق الكود سينتج الشكل التالي حيث قلت حدة اللون الأول لأنه بدأ قبل الزاوية 0deg وقلت حدة اللون الأخير لأنه انتهى بعد الزاوية 360deg:

### تغيير خط نشأة التدرج

يمكن تحديد خط نشأة أو بداية التدرج المخروطي بتحديد الزاوية التي يبدأ عندها التدرج المخروطي باستخدام المعامل form ثم وضع الزاوية بعدها كالتالي:

```
background-image: conic-gradient(from 0deg, pink, red 35%,
green 60%, yellow 85%, blue);
```



from 0deg

from 45deg

from 90deg

from 210deg

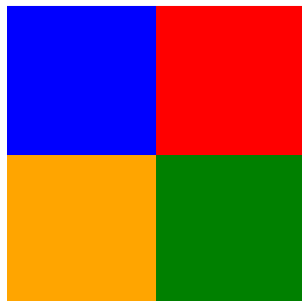
from 300deg

### إلغاء التداخل بين الألوان

يمكن أن نلغي المساحة المتداخلة بين كل لونين ونجعل كل لون خالص بدون

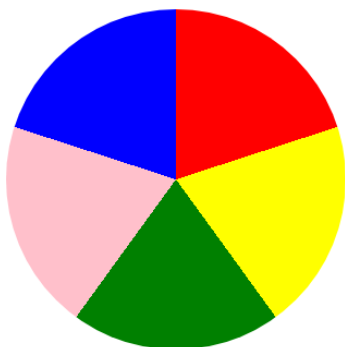
أي تداخل مع اللون التالي وذلك بجعل نقطة نهاية اللون هي نقطة بداية اللون الذي يليه كالتالي:

**background-image:** conic-gradient(red 25%, green 25% 50%, orange 50% 75%, blue 75%);



**border-radius:** 50%;

**background-image:** conic-gradient(red 0.2turn, yellow 0.2turn 0.4turn, green 0.4turn 0.6turn, pink 0.6turn 0.8turn, blue 0.8turn );

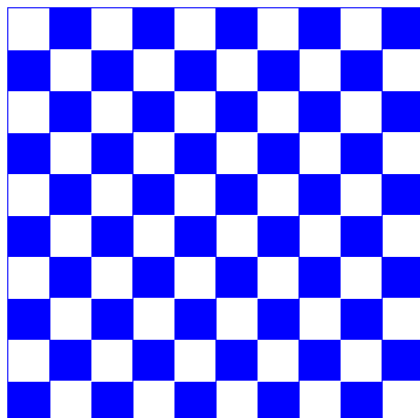


لاحظ أننا استخدمنا وحدة قياس الزوايا turn بدلاً من deg ، أننا استخدمنا الخاصية border-radius للتحويل المربع إلى دائرة.

مثال آخر:

**border:** 0.5px solid blue;

**background:** conic-gradient(blue 25%, white 25% 50%, blue 50% 75%, white 75%) left top/20% 20%;

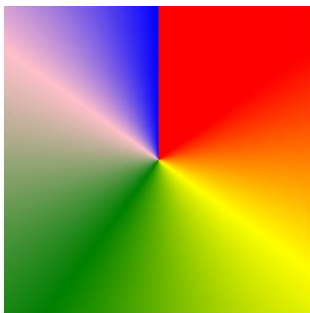


## تحديد موقع مركز التدرج

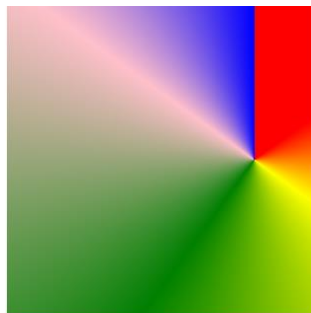
يمكن أن يكون مركز التدرج في موقع آخر غير مركز العنصر ويتم ذلك بتعيين المعامل الأول لدالة conic-gradient بالمعامل at ويتبعه بعض القيم كالتالي:

1- نسبة المسافة الأفقية بين مركز التدرج والحد الأيسر للعنصر إلى عرض العنصر ورأسياً عن طريق نسبة المسافة الرأسية بين مركز التدرج والحد العلوي للعنصر إلى ارتفاع العنصر كالتالي:

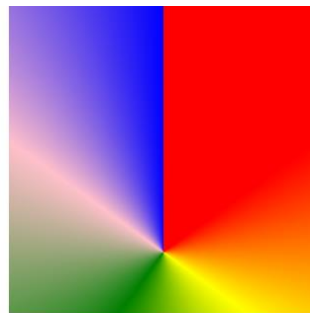
```
background-image: conic-gradient(at 50% 50%, red 15%,  
yellow 35%,green 60%, pink 85%,blue);
```



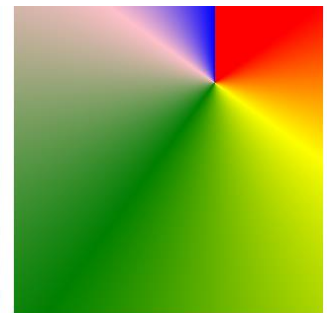
at 50% 50%



at 80% 50%



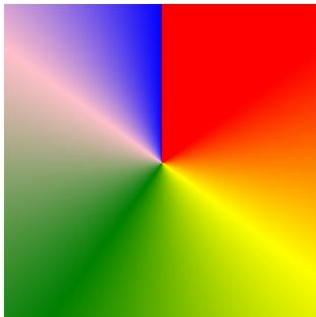
at 50% 80%



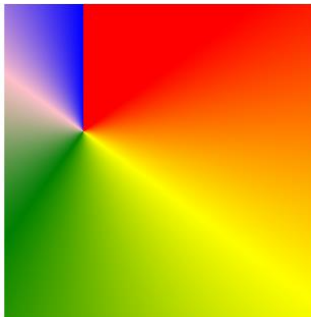
at 65% 25%

2- الإزاحة الأفقية كقيمة مطلقة بين مركز التدرج ونقطة بداية القطر الأفقي للتدرج (الحد الأيسر للعنصر)، والإزاحة الرأسية كقيمة مطلقة بين مركز التدرج ونقطة بداية القطر الرأسي للتدرج (الحد العلوي للعنصر).

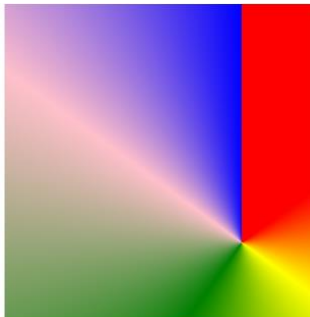
```
background-image: conic-gradient(at 100px 100px, red 15%,  
yellow 35%,green 60%, pink 85%,blue);
```



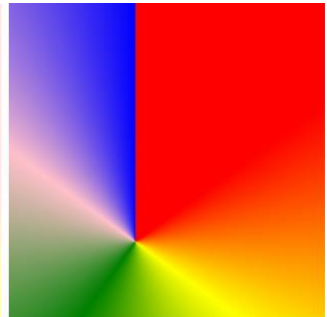
at 100px 100px



at 50px 80px



at 150px 150px



at 80px 150px

3- قيم ثابتة وهي كالتالي:

- at top : وتعني أن مركز التدرج في منتصف الحد العلوي للعنصر أي أن نسبة الإزاحة الأفقية تساوي 50% من القطر الأفقي للتدرج، بينما نسبة الإزاحة الرأسية

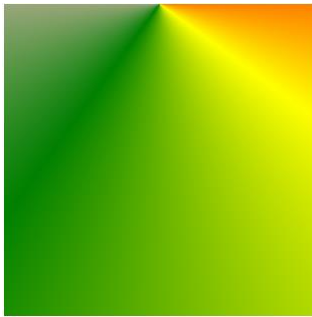
تساوي 0% من القطر الرأسي للتدرج.

- **at right** : وتعني أن مركز التدرج في منتصف الحد الأيمن للعنصر أي أن نسبة الإزاحة الأفقية تساوي 100% من القطر الأفقي للتدرج، بينما نسبة الإزاحة الرأسية تساوي 50% من القطر الرأسي للتدرج.

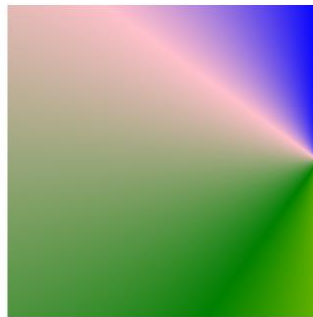
- **at bottom** : وتعني أن مركز التدرج في منتصف الحد السفلي للعنصر أي أن نسبة الإزاحة الأفقية تساوي 50% من القطر الأفقي للتدرج، بينما نسبة الإزاحة الرأسية تساوي 100% من القطر الرأسي للتدرج.

- **at left** : وتعني أن مركز التدرج منتصف الحد الأيسر للعنصر أي أن نسبة الإزاحة الأفقية تساوي 0% من القطر الأفقي للتدرج، بينما نسبة الإزاحة الرأسية تساوي 50% من القطر الرأسي للتدرج.

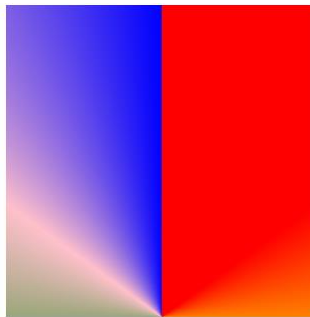
**background-image: conic-gradient(at top, red 15%, yellow 35%, green 60%, pink 85%, blue);**



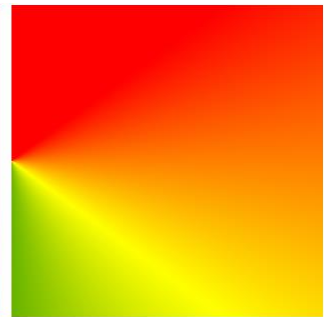
**at top**



**at right**



**at bottom**



**at left**

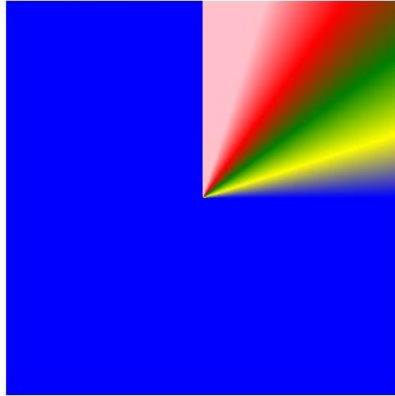
قم بتغيير القيم كما بالشكل ولاحظ اختلاف موقع مركز التدرج في كل مرة.



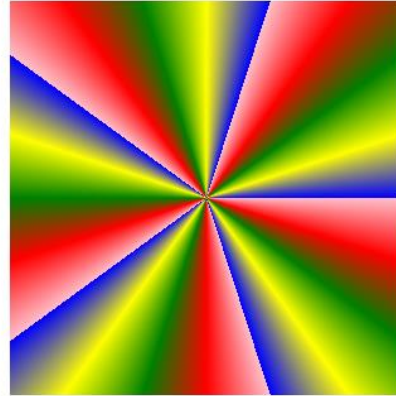
**repeating-conic-gradient**

ويتكرر التدرج المخروطي وذلك بتعيين كل مساحة التدرج في نسبة من محيط التدرج تساوي 100% مقسوماً على عدد مرات التكرار المطلوبة كالتالي:

**background-image: repeating-conic-gradient(pink 5%, red 10%, green 15%, yellow 20%, blue 25%);**



conic-gradient



repeating-conic-gradient

لاحظ أن التدرج المخروطي انتهى عند نقطة تساوي 20% من نصف قطر التدرج، ولذلك فإن اللون الأخير سيملاً باقي مساحة خلفية العنصر (حوالي 80%) كما بالشكل الأول.

لاحظ أن التدرج المخروطي المتكرر اعتبر أن التدرج قد انتهى عند النقطة 20% ولذلك لم يبق بملاء باقي المساحة بآخر لون في التدرج، ولكنه قام بتكرار التدرج في باقي المساحة (80%) بنفس النسبة (20%)، فنشأ لدينا 4 تدرجات جديدة بالإضافة للتدرج الأصلي، أي أصبحت خلفية العنصر مملوءة بتدرج واحد مكرر خمس مرات.

**mix-blend-mode**

خاصية تحدد كيف يتم دمج محتوى عنصر أمامي مع محتوى عنصر آخر خلفي، حيث يسمى العنصر الأمامي source color أو لون المصدر، ويسمى العنصر الخلفي backdrop color أو لون التقاطع، وقد يكون العنصر الأمامي هو عنصر الابن والعنصر الخلفي هو العنصر الأب أو العنصر body، والخاصية غير وراثية وتقبل عدة أنواع من القيم وهي نفس قيم الخاصية background-blend-mode كالتالي:

**ملف html**

```
<div class="black">Mix Blend<main class="white">Mode
</main></div>
```

**ملف css**

```
body{
  font: 800 36px arial;
  background:url(blue.jpg) center/100%;
}
div{
  width: 200px;
  height: 120px;
  background: linear-gradient(white , black) ;
  text-align: center;
  margin-left: 20px;
  padding-top:15px;
  box-sizing: border-box;
  mix-blend-mode: normal;
}
.black{
  color: white;
}
.white{
  color: black;
  line-height: 50px;
}
```



backdrop color



source color

لاحظ أننا فصلنا اللونين لتوضيح أن لكل لون منهم خلفية مختلفة وعند التطبيق سيكون اللون الأمامي هو لون المصدر source واللون الخلفي هو لون التقاطع backdrop.

وسوف نستعرض قيم الخاصية بشيء من الاختصار حيث شرحناها جميعاً بالتفصيل عند حديثنا عن قيم الخاصية background-blend-mode كما ذكرنا من قبل.

- **normal** : القيمة الافتراضية ولا يحدث أي دمج أو تأثير على الألوان.

```
mix-blend-mode: normal;
```



- **multiply** : قيمة تعني أن اللون الناتج

يكون حاصل ضرب القيمة اللونية للون المصدر والقيمة اللونية للون التقاطع، وبناء على ذلك فإن أي لون يتقاطع مع اللون الأسود يكون الناتج لون أسود، وأي لون

يتقاطع مع اللون الأبيض يكون الناتج هو نفس اللون:

```
mix-blend-mode: multiply;
```



- **screen** : قيمة تعني أن حجب لون

المصدر بلون تقاطع أبيض ينتج عنه لون مصدر أبيض، بينما حجب لون المصدر بلون متقاطع أسود ينتج عنه نفس اللون.



`mix-blend-mode: screen;`



- **overlay** : قيمة تعني اختلاط لون التقاطع مع لون المصدر لينعكس لون التقاطع من لون المصدر:

`mix-blend-mode: overlay;`



- **darken** : قيمة تعني أن لون التقاطع يستبدل لون المصدر إذا كان أذكى منه، أما إذا كان لون التقاطع أفتح فيحتفظ لون المصدر بلونه.

`mix-blend-mode: darken;`



- **lighten** : قيمة تعني أن لون التقاطع يستبدل لون المصدر إذا كان أفتح منه، أما إذا كان لون التقاطع أذكى فيحتفظ لون المصدر بلونه.

`mix-blend-mode: lighten;`



- **color-dodge** : قيمة تعني أن ألوان التقاطع تلون كل ألوان المصدر، وتجعلها أكثر سطوعاً، ما عدا اللون الأسود فلا يظهر عليه تأثير السطوع.

`mix-blend-mode: color-dodge;`



- **color-burn** : قيمة تعني أن لون التقاطع يلون كل ألوان المصدر وتجعل ألوان المصدر أكثر دكاسة، ما عدا اللون الأبيض فلا يظهر عليه تأثير الدكاسة.

**mix-blend-mode: color-burn;**



- **hard-light** : قيمة تعني أن لون المصدر مسلط عليه بقعة ضوء قوية من نفس لون التقاطع.

**mix-blend-mode: hard-light;**



- **soft-light** : قيمة تعني أن لون المصدر مسلط عليه بقعة ضوء أكثر انتشاراً وأقل قوة من نفس لون التقاطع.

**mix-blend-mode: soft-light;**



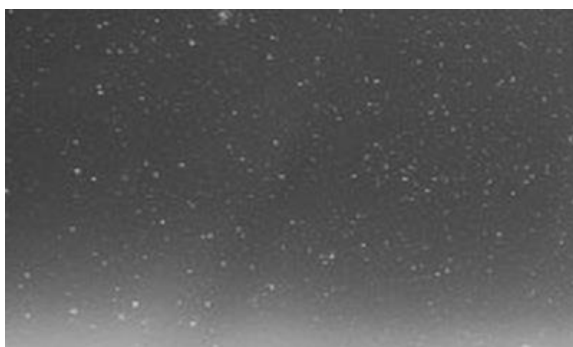
- **difference** : قيمة تعني أن لون المصدر يحذف منه اللون الأكثر دكاسة من لون التقاطع، فإذا كان لون التقاطع أبيض، فيتم عكس لون المصدر، أما لون التقاطع الأسود فلا يظهر أي تأثير على لون المصدر.

**mix-blend-mode: difference;**



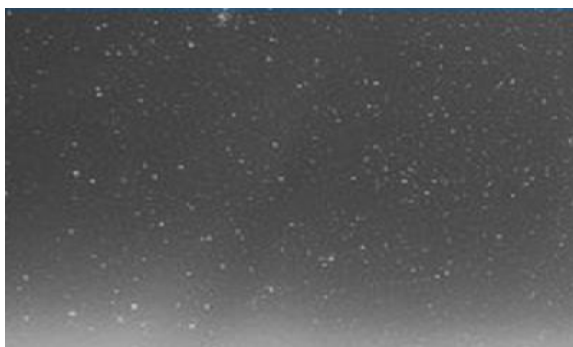
- **exclusion** : قيمة تشبة القيمة difference ولكن لون المصدر يكون أقل تبايناً contrast ونفس تأثير الانعكاس مع اللون الأبيض أما اللون الأسود فلا تأثير له.

**mix-blend-mode: exclusion;**



- **hue** : قيمة تعني أن درجة اللون hue تكون من لون المصدر وأن نسبة التشبع saturation و نسبة الإضاءة luminosity تكون من لون التقاطع.

**mix-blend-mode: hue;**



- **saturation** : قيمة تعني أن نسبة تشبع اللون saturation تكون من لون المصدر وأن درجة اللون hue ونسبة الإضاءة luminosity تكون من لون التقاطع.

**mix-blend-mode: sturation;**



- **luminosity** : قيمة تعني أن نسبة الإضاءة luminosity تكون من لون المصدر وأن درجة اللون hue ونسبة التشبع saturation تكون من لون التقاطع.

**mix-blend-mode: luminosity;**

## filter

خاصية تقوم بعمل تأثيرات لونية وبصرية وظلال وشفافية على العنصر الذي في الأغلب يكون صورة image، والخاصية غير وراثية ولها عدة قيم كالتالي:

## ملف html

```

```

## ملف css

```
img{
  filter: drop-shadow(3px 3px 3px blue);
}
```

- **drop-shadow** : دالة تقوم بإسقاط ظل للعنصر والفارق بينها وبين خاصية box-shadow أنها تراعي الشفافية في خلفية العنصر وبين أجزائه مثل الصور ذات تنسيق png، بينما خاصية box-shadow لا تراعي الشفافية وتسقط الظل حسب عرض وارتفاع العنصر.

المعاملات داخل قوسي الدالة drop-shadow هي نفس معاملات الخاصيتين box-shadow و text-shadow ولها نفس السلوك والقيم الافتراضية والقيم الموجبة والسالبة، ويمكن الرجوع لشرح الخاصيتين لمزيد من التفاصيل.



original



box-shadow



drop-shadow

قد يقبل العنصر (الصورة) أكثر من دالة لإسقاط ظل له من أكثر من زاوية ويفصل بين دالة الظل والتي تليها بمسافة فارغة كالتالي:

```
img{
  filter: drop-shadow( 3px  3px 3px blue)
         drop-shadow(-3px  3px 3px blue)
```



```
drop-shadow( 3px -3px 3px blue)
drop-shadow(-3px -3px 3px blue);
```

```
}
```



لاحظ أن تغيير إشارة قيمة الإزاحة الأفقية أو الرأسية يعكس اتجاه الظل، وأن عدم تعيين قيمة لتشتت الظل في كل الدوال أو إعطاءها قيمة تساوي 0px يظهر الظل وكأنه حدود خارجية للعنصر.

**blur** - دالة تظهر الصورة مطموسة المعالم والمعامل داخل قوسي الدالة blur عبارة عن عدد النقاط اللونية pixels التي يتم دمجها مع بعضها لتظهر الصورة مطموسة، وكلما زاد عدد النقاط اللونية زاد معدل طمس الصورة كالتالي:

```
img{
  filter: blur(2px);
}
```



original



2px



7px

**brightness** - دالة تحدد مقدار الإضاءة الساقطة على العنصر (الصورة) وهي تقدر كنسبة مئوية وتتراوح بين 0% ومضاعفات 100% والتي تعتبر نسبة الإضاءة الافتراضية، ويمكن أن تكون القيمة مضاعفات 1 أو جزء منه، فالقيمة 1.5 تعني 150% والقيمة 0.75 تعني 75% والقيمة 0 تعني 0% وتعني أن العنصر

```
img{
  filter: brightness(40%);
}
```



- **contrast** : دالة تحدد مقدار التباين في ألوان العنصر (الصورة) وهي كالنسبة السابقة تقدر كنسبة مئوية أو مضاعفات 100% أو كجزء من 1 أو مضاعفاته فالنسبة الافتراضية للتباين هي 1 أو 100% والنسبة 0% تعني عدم وجود تباين بين الألوان وبالتالي يكون لون العنصر رمادياً كالتالي:

```
img{
  filter: contrast(40%);
}
```



- **grayscale** : دالة تحول ألوان العنصر (الصورة) إلى تدرج اللون الرمادي، وهي كالقيمتين السابقتين، تقبل نسبة مئوية أو جزء من 1 ولكنها لا تقبل قيمة أكثر من 100% أو أكثر من 1 والنسبة الافتراضية للون العنصر الأصلي هي 0% بينما النسبة 100% تعني أن لون العنصر أصبح تدرج لوني رمادي كالتالي:

```
img{
```

```
filter: grayscale(50%);
}
```



original



50%



100%

- **invert** : دالة تعني عكس الألوان على دائرة الألوان، وتقبل نسبة مئوية أو جزء من 1 ولا تقبل قيمة أكثر من 100% وهي تساوي 1 والنسبة الافتراضية للون العنصر الأصلي هي 0% بينما النسبة 0% تعني أن لون العنصر أصبح منعكساً كلية كالتالي:

```
img{
  filter: invert(50%);
}
```



original



25%



0.5



100%

- **opacity** : دالة تتحكم في شفافية العنصر (الصورة)، وتقبل نسبة مئوية أو جزء من 1 ولا تقبل قيمة أكثر من 100% وهي تساوي 1 والنسبة الافتراضية لشفافية العنصر الأصلي هي 100% بينما النسبة 0% تعني أن العنصر أصبح شفافاً كلية كالتالي:

```
img{
```

```
filter: opacity(50%);
}
```



original



75%



50%



0.2

- **saturate** : دالة نسبة تشبع ألوان العنصر، وتقبل نسبة مئوية أو جزء من 1 ولا تقبل قيمة أكثر من 100% وهي تساوي 1 والنسبة الافتراضية لتشبع لون العنصر الأصلي هي 100% بينما النسبة 0% تعني أن لون العنصر أصبح تدرجاً لونياً رمادياً كالتالي:

```
img{
  filter: saturate(50%);
}
```



original



60%



0.4



0%

- **sepia** : دالة تعني ألوان العنصر تتحول إلى تدرج لوني بني داكن، وتقبل نسبة مئوية أو جزء من 1 ولا تقبل قيمة أكثر من 100% وهي تساوي 1 والنسبة الافتراضية للون العنصر الأصلي هي 0% بينما النسبة 100% تعني أن لون العنصر أصبح تدرج بني داكن تماماً (تأثير الأوراق القديمة) كالتالي:

```
img{
  filter: sepia(60%);
}
```





original



60%



100%

- **hue-rotate** : دالة تعني أن لون العنصر يصبح درجة معينة على دائرة الألوان وبناء على ذلك فالقيمة تكون زاوية بأي وحدة قياس مما سبق دراستها، وتكون بين 0deg و 360deg أو بين 0turn و 1turn أو غيرها من الوحدات كالتالي:

```
img{
  filter: hue-rotate(90deg);
}
```



original



90deg



0.55turn



300grad

ويمكن للعنثر أن يقبل أكثر من قيمة لخاصية filter ليقع على العنصر أكثر من تأثير على أن يفصل بين كل قيمة والتي تليها بمسافة فارغة كالتالي:

```
img{
  filter: drop-shadow(3px 3px 3px) grayscale(1) blur(1px);
}
```

**original****filters**

لاحظ أن العنصر (الصورة) تأثر بثلاث قيم للخاصية **filter** القيمة الأولى أسقطت ظل للعنصر ولقيمة الثانية حولت الألوان إلى تدرج لوني رمادي، والثالثة قامت بطمس معالم العنصر، ويمكن إضافة قيمة رابعة وخامسة وأكثر حسب التنسيق المطلوب.