

CSS BOX SIZING

KÍCH THƯỚC HỘP TRONG CSS

Thuộc tính box-sizing trong CSS cho phép chúng tôi bao gồm phần đệm padding và đường viền border trong tổng chiều rộng và chiều cao của phần tử.

KHI KHÔNG SỬ DỤNG THUỘC TÍNH CSS BOX-SIZING

Theo mặc định, chiều rộng và chiều cao của một phần tử được tính như sau:

$\text{width} + \text{padding} + \text{border} = \text{chiều rộng thực tế của 1 thẻ html}$

$\text{height} + \text{padding} + \text{border} = \text{chiều cao thực tế của 1 thẻ html}$

Điều này có nghĩa là: Khi bạn đặt chiều rộng / chiều cao của một phần tử, phần tử thường xuất hiện lớn hơn bạn đã đặt (vì đường viền và phần đệm của phần tử được thêm vào chiều rộng / chiều cao được chỉ định của phần tử).

Hình minh họa sau đây cho thấy hai phần tử `<div>` có cùng chiều rộng và chiều cao được chỉ định:

Div này nhỏ hơn (chiều rộng là 300px và chiều cao là 100px).

Div này lớn hơn (chiều rộng cũng là 300px và chiều cao là 100px).

Hai phần tử `<div>` ở trên kết thúc với các kích thước khác nhau trong kết quả (vì `div2` có một phần đệm được chỉ định);

```
.div1 {  
    width: 300px;  
    height: 100px;  
    border: 1px solid blue;  
}  
  
.div2 {  
    width: 300px;  
    height: 100px;  
    padding: 50px;  
    border: 1px solid red;  
}
```

Thuộc tính box-sizing sẽ giải quyết vấn đề này .

KHI CÓ SỬ DỤNG THUỘC TÍNH CSS BOX-SIZING

Các thuộc tính box-sizing cho phép chúng ta bao gồm padding và border trong tổng chiều rộng và chiều cao của một thẻ html

.

Nếu bạn đặt box-sizing: border-box; trên một phần tử, phần đệm padding và đường viền border sẽ được bao gồm trong chiều rộng và chiều cao của thẻ html đó :

Cả hai div đều có cùng kích thước bây giờ!

Hoan hô!

Đây là ví dụ tương tự như trên, box-sizing: border-box; được thêm vào cả hai phần tử <div>:

```
.div1 {  
    width: 300px;  
    height: 100px;  
    border: 1px solid blue;  
    box-sizing: border-box;  
}  
  
.div2 {  
    width: 300px;  
    height: 100px;  
    padding: 50px;  
    border: 1px solid red;  
    box-sizing: border-box;  
}
```

Vì kết quả của việc sử dụng

box-sizing: border-box;

tốt hơn rất nhiều, nhiều nhà phát triển muốn tất cả các phần tử trên trang của họ hoạt động theo cách này.

Đoạn mã dưới đây đảm bảo rằng tất cả các phần tử đều có kích thước theo cách trực quan hơn. Nhiều trình duyệt đã sử dụng

box-sizing: border-box;

cho nhiều phần tử biểu mẫu form (nhưng không phải tất cả - đó là lý do tại sao inputs và textarea trông khác nhau khi chiều rộng: 100%;).

Áp dụng điều này cho tất cả các yếu tố là an toàn và khôn ngoan:

```
* {  
    box-sizing: border-box;  
}
```

CSS BOX SHADOW

CSS BOX SHADOW

Thuộc tính CSS box-shadow áp dụng đổ bóng cho các phần tử thẻ html .

Trong cách sử dụng đơn giản nhất, bạn chỉ xác định bóng ngang và bóng dọc:

Đây là phần tử <div> màu vàng
với bóng hộp màu đen

```
div {  
  box-shadow: 10px 10px;  
}
```

Tiếp theo, thêm màu vào bóng:

Đây là phần tử <div> màu vàng
với bóng hộp màu xám

```
div {  
    box-shadow: 10px 10px grey;  
}
```

Tiếp theo, thêm hiệu ứng mờ vào bóng:

Đây là phần tử <div> màu vàng
với bóng hộp màu xám, mờ

```
div {  
    box-shadow: 10px 10px 5px grey;  
}
```

CÚ PHÁP :

box-shadow: A B C D;

A là đổ bóng theo chiều ngang

B là đổ bóng theo chiều dọc

C là pixel làm mờ khi đổ bóng

D là màu đổ bóng

ví dụ :

box-shadow: 10px 10px 5px grey;

CSS TRANSITIONS

Chuyển tiếp CSS cho phép bạn thay đổi các giá trị thuộc tính một cách trơn tru, trong một khoảng thời gian nhất định.

Di chuột qua phần tử bên dưới để xem hiệu ứng chuyển tiếp CSS:

Trong chương này, bạn sẽ tìm hiểu về các thuộc tính sau:

- transition
- transition-delay
- transition-duration
- transition-property
- transition-timing-function

LÀM THẾ NÀO ĐỂ SỬ DỤNG TRANSITION CHUYỂN TIẾP TRONG CSS?

Để tạo hiệu ứng chuyển tiếp, bạn phải chỉ định hai điều:

- thuộc tính CSS bạn muốn thêm hiệu ứng vào
- thời gian của hiệu ứng

Lưu ý: Nếu phần thời lượng không được chỉ định, quá trình chuyển đổi sẽ không có hiệu lực, vì giá trị mặc định là 0.

Ví dụ sau đây cho thấy phần tử `<div>` màu đỏ `100px * 100px`. Phần tử `<div>` cũng đã chỉ định hiệu ứng chuyển tiếp cho thuộc tính `width`, với thời lượng 2 giây:

```
div {  
    width: 100px;  
    height: 100px;  
    background: red;  
    transition: width 2s;  
}
```

Hiệu ứng chuyển tiếp sẽ bắt đầu khi thuộc tính CSS được chỉ định (chiều rộng) thay đổi giá trị.

Bây giờ, hãy để chúng tôi chỉ định một giá trị mới cho thuộc tính width khi người dùng di chuột qua phần tử <div>:

```
div:hover {  
    width: 300px;  
}
```

Lưu ý rằng khi con trỏ di chuyển ra khỏi phần tử, nó sẽ dần dần thay đổi trở lại kiểu ban đầu.

THAY ĐỔI MỘT SỐ GIÁ TRỊ THUỘC TÍNH

Ví dụ sau đây thêm hiệu ứng chuyển tiếp cho cả thuộc tính width và height, với thời lượng 2 giây cho chiều rộng và 4 giây cho chiều cao:

```
div {  
    transition: width 2s, height 4s;  
}
```

CHỈ ĐỊNH ĐƯỜNG CONG TỐC ĐỘ CỦA QUÁ TRÌNH CHUYỂN ĐỔI

Các transition-timing-functionbất động sản quy định các đường cong tốc độ của hiệu ứng chuyển tiếp.

Thuộc tính chức năng thời gian chuyển tiếp có thể có các giá trị sau:

- ease - chỉ định hiệu ứng chuyển tiếp với bắt đầu chậm, sau đó nhanh, sau đó kết thúc chậm (đây là mặc định)
- linear - chỉ định một hiệu ứng chuyển tiếp với cùng tốc độ từ đầu đến cuối
- ease-in - chỉ định hiệu ứng chuyển tiếp với khởi động chậm
- ease-out - chỉ định hiệu ứng chuyển tiếp với kết thúc chậm
- ease-in-out - chỉ định hiệu ứng chuyển tiếp với bắt đầu và kết thúc chậm
- cubic-bezier(n,n,n,n) - cho phép bạn xác định các giá trị của riêng mình trong một hàm bậc ba

Ví dụ sau đây cho thấy một số đường cong tốc độ khác nhau có thể được sử dụng:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
```

width: 100px;

height: 100px;

background: red;

```
    transition: width 2s;
}

#div1 {transition-timing-function: linear;}
#div2 {transition-timing-function: ease;}
#div3 {transition-timing-function: ease-in;}
#div4 {transition-timing-function: ease-out;}
#div5 {transition-timing-function: ease-in-out;}
```

```
div:hover {
    width: 300px;
}
</style>
</head>
<body>
```

<h1>The transition-timing-function Property</h1>

<p>Hover over the div elements below, to see the different speed curves:</p>

```
<div id="div1">linear</div><br>
<div id="div2">ease</div><br>
<div id="div3">ease-in</div><br>
<div id="div4">ease-out</div><br>
<div id="div5">ease-in-out</div>
```

```
<div><div>ease-in-out</div><br>

<p><b>Note:</b> This example does not work in Internet Explorer 9 and earlier versions.</p>

</body>
</html>
```

Xem ví dụ tại đây :

https://www.w3schools.com/css/tryit.asp?filename=trycss3_transition_speed

TRÌ HOÃN HIỆU ỨNG CHUYỂN ĐỔI

Các thuộc tính transition-delay quy định cụ thể một sự chậm trễ (tính bằng giây) cho các hiệu ứng chuyển.

Ví dụ sau có độ trễ 1 giây trước khi bắt đầu:

```
div {  
    transition-delay: 1s;  
}
```

Ví dụ không sử dụng cách viết tắt :

```
div {  
    transition-property: width;  
    transition-duration: 2s;  
    transition-timing-function: linear;  
    transition-delay: 1s;  
}
```

Sử dụng cách viết tắt :

```
div {  
  transition: width 2s linear 1s;  
}  
 
```

CSS TRANSFORMS

CSS 2D TRANSFORMS

Các phép biến đổi CSS cho phép bạn di chuyển, xoay, chia tỷ lệ và xiên các phần tử.

2D

rotate

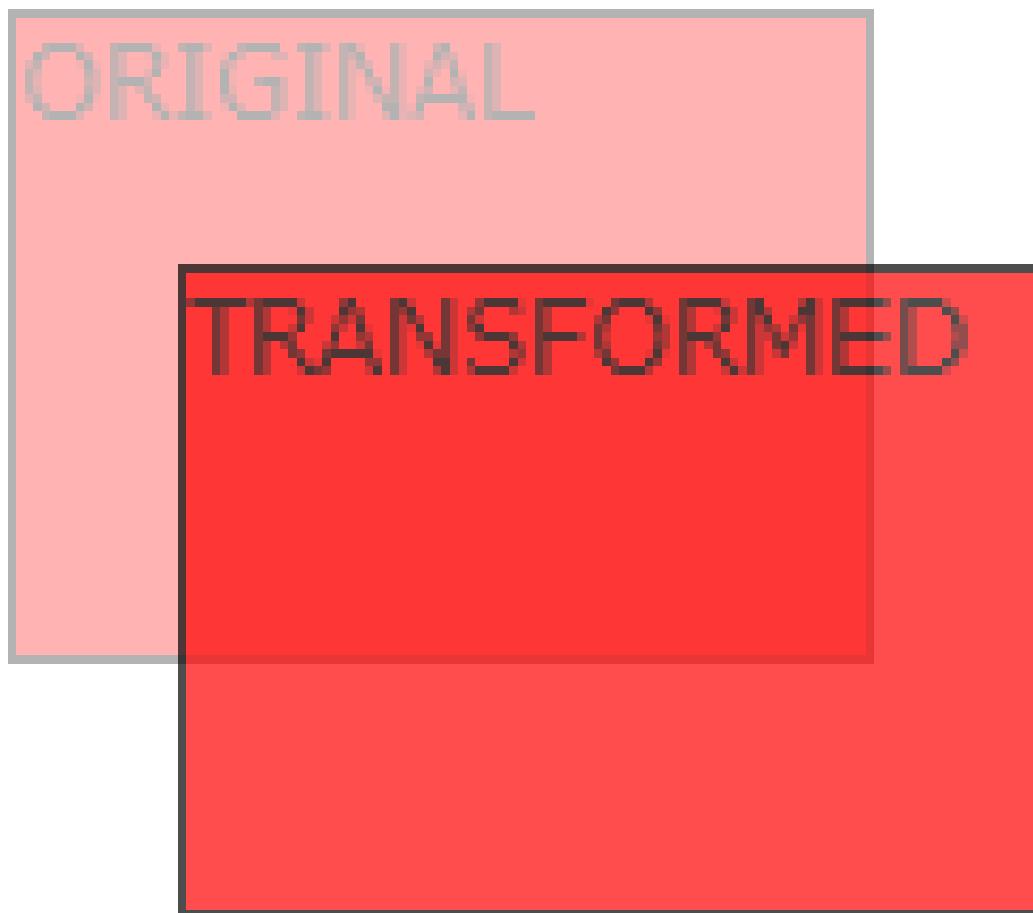


CSS 2D TRANSFORMS METHODS

các phương pháp transforms

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

The translate() Method



Các phương pháp `translate()` di chuyển một phần tử từ vị trí hiện tại của mình (theo các thông số đưa ra cho các trục X và trục Y).

Ví dụ sau di chuyển phần tử `<div>` 50 pixel sang bên phải và 100 pixel xuống từ vị trí hiện tại của nó:

```
div {  
    transform: translate(50px, 100px);  
}
```

The rotate() Method



Các phương pháp `rotate()` quay một chiều kim đồng hồ nguyên tố hoặc truy cập chiều kim đồng hồ theo một mức độ nhất định.

Ví dụ sau xoay phần tử `<div>` theo chiều kim đồng hồ với 20 độ:

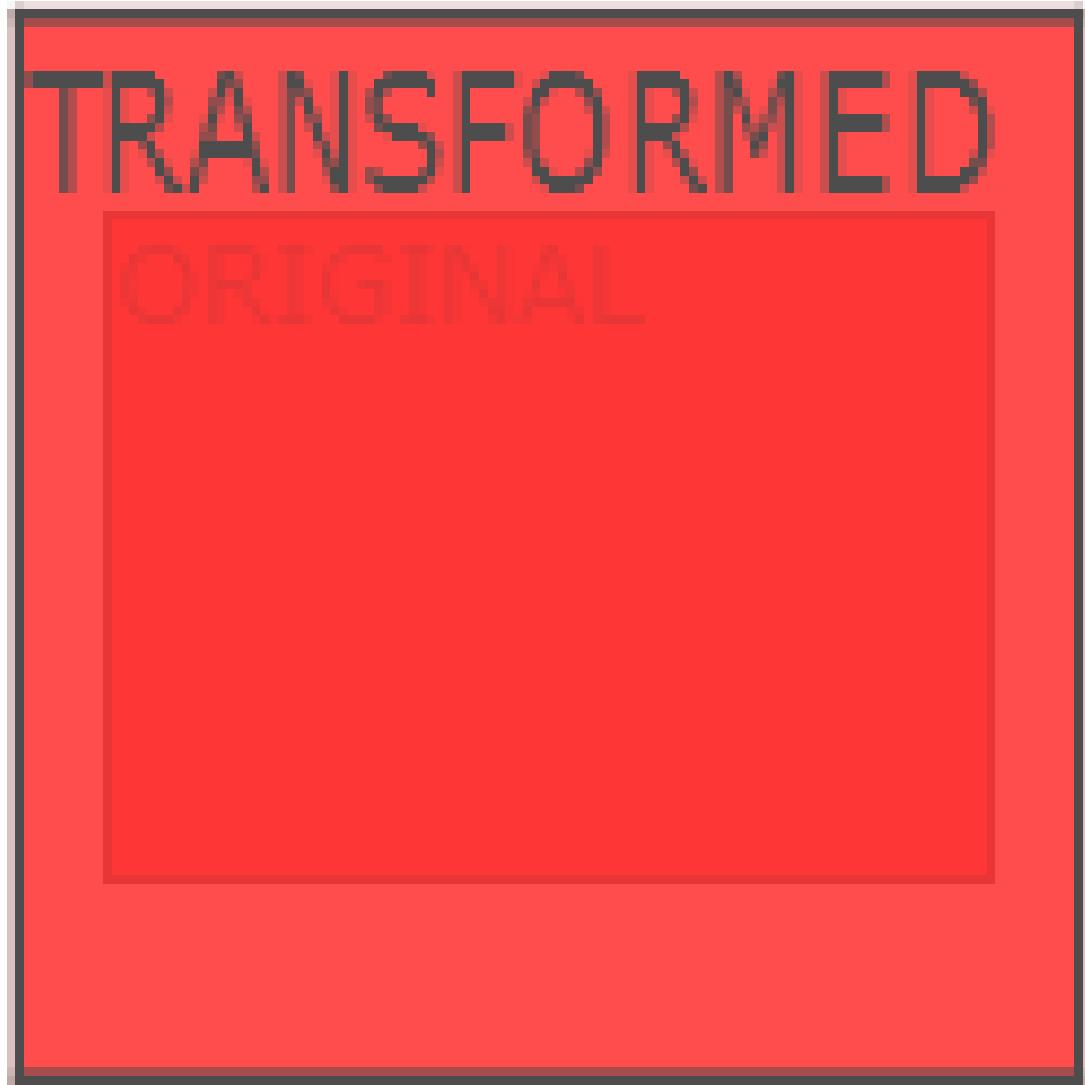
```
div {  
    transform: rotate(20deg);  
}
```

Sử dụng các giá trị âm sẽ xoay phần tử ngược chiều kim đồng hồ.

Ví dụ sau xoay phần tử <div> ngược chiều kim đồng hồ với 20 độ:

```
div {  
    transform: rotate(-20deg);  
}
```

Phương thức scale()



Các phương thức `scale()` tăng hoặc giảm kích thước của một phần tử (theo các thông số đưa ra cho chiều rộng và chiều cao).

Ví dụ sau tăng phần tử `<div>` lên hai lần chiều rộng ban đầu và ba lần chiều cao ban đầu:

Thí dụ

```
div {  
  transform: scale(2, 3);  
}
```

Ví dụ sau giảm phần tử `<div>` còn một nửa chiều rộng và chiều cao ban đầu của nó:

Thí dụ

```
div {  
    transform: scale(0.5, 0.5);  
}
```

Phương thức scaleX()

Các phương thức `scaleX()` tăng hoặc giảm chiều rộng của một phần tử.

Ví dụ sau đây tăng phần tử `<div>` lên hai lần chiều rộng ban đầu của nó:

Thí dụ

```
div {  
    transform: scaleX(2);  
}
```

Ví dụ sau giảm phần tử `<div>` xuống còn một nửa chiều rộng ban đầu của nó:

Thí dụ

```
div {  
    transform: scaleX(0.5);  
}
```

Phương thức scaleY()

Các phương thức `scaleY()` tăng hoặc giảm chiều cao của một phần tử.

Ví dụ sau tăng phần tử `<div>` lên ba lần chiều cao ban đầu của nó:

Thí dụ

```
div {  
    transform: scaleY(3);  
}
```

Ví dụ sau giảm phần tử `<div>` xuống còn một nửa chiều cao ban đầu của nó:

Thí dụ

```
div {  
    transform: scaleY(0.5);  
}
```

Phương thức skewX()

Các phương thức `skewX()` làm lệch một yếu tố dọc theo trục X bởi các góc độ nhất định.

Ví dụ sau làm nghiêng phần tử `<div>` 20 độ dọc theo trục X:

```
div {  
    transform: skewX(20deg);  
}
```

Phương thức skewY()

Các phương thức `skewY()` làm lệch một yếu tố dọc theo trục Y theo góc độ nhất định.

Ví dụ sau làm nghiêng phần tử `<div>` 20 độ dọc theo trục Y:

```
div {  
    transform: skewY(20deg);  
}
```

Phương thức skew()

Các phương pháp skew() làm lệch một yếu tố dọc theo X và trục Y bởi những góc nhất định.

Ví dụ sau làm nghiêng phần tử <div> 20 độ dọc theo trục X và 10 độ dọc theo trục Y:

Thí dụ

```
div {  
    transform: skew(20deg, 10deg);  
}
```

Nếu tham số thứ hai không được chỉ định, nó có giá trị bằng không. Vì vậy, ví dụ sau làm nghiêng phần tử <div> 20 độ dọc theo trục X:

Thí dụ

```
div {  
    transform: skew(20deg);  
}
```

Phương thức matrix()

Các phương thức matrix() kết hợp tất cả phương pháp 2D transform thành một.

Phương thức matrix () nhận sáu tham số, chứa các hàm toán học, cho phép bạn xoay, chia tỷ lệ, di chuyển (dịch) và xiên các phần tử.

Các tham số như sau:

matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

```
div {  
    transform: matrix(1, -0.3, 0, 1, 0, 0);  
}
```

CSS

CSS GRADIENT

CSS GRADIENTS

Nền Gradient

CSS gradient cho phép bạn hiển thị các chuyển tiếp mượt mà giữa hai hoặc nhiều màu được chỉ định.

CSS xác định hai loại gradient:

- **Gradients tuyến tính (đi xuống / lên / trái / phải / theo đường chéo)**
- **Radial Gradients (được xác định bởi trung tâm của chúng)**

CSS LINEAR GRADIENTS

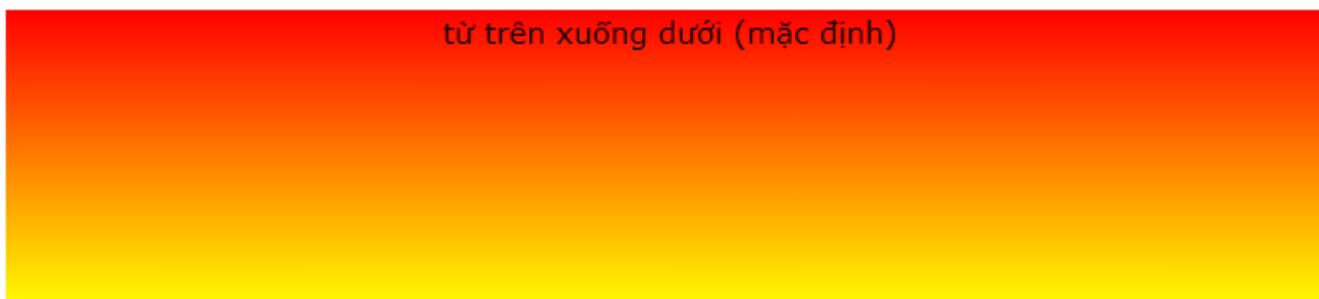
Để tạo một gradient tuyến tính, bạn phải xác định ít nhất hai điểm dừng màu. Điểm dừng màu là những màu bạn muốn tạo ra các chuyển tiếp mượt mà. Bạn cũng có thể đặt điểm bắt đầu và hướng (hoặc góc) cùng với hiệu ứng gradient.

Cú pháp

```
background-image: linear-gradient(direction, color-stop1, color-stop2, ...);
```

Hướng - Từ trên xuống dưới (đây là mặc định)

Ví dụ sau đây cho thấy một gradient tuyến tính bắt đầu ở trên cùng. Nó bắt đầu màu đỏ, chuyển sang màu vàng:



Ví dụ minh họa :

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 200px;
    background-color: red; /* For browsers that do not
support gradients */
    background-image: linear-gradient(red, yellow);
}
</style>
</head>
<body>

<h1>Linear Gradient - Top to Bottom</h1>
<p>This linear gradient starts red at the top,
transitioning to yellow at the bottom:</p>

<div id="grad1"></div>

</body>
</html>
```

Linear Gradient - Top to Bottom

This linear gradient starts red at the top, transitioning to yellow at the bottom:



Hướng - Trái sang phải

Ví dụ sau đây cho thấy một gradient tuyến tính bắt đầu từ bên trái. Nó bắt đầu màu đỏ, chuyển sang màu vàng:

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 200px;
    background-color: red; /* For browsers that do not
support gradients */
    background-image: linear-gradient(to right, red , yellow);
}
</style>
</head>
<body>

<h1>Linear Gradient - Left to Right</h1>
<p>This linear gradient starts red at the left,
transitioning to yellow (to the right):</p>

<div id="grad1"></div>

</body>
</html>
```

Linear Gradient - Left to Right

This linear gradient starts red at the left, transitioning to yellow (to the right):



Hướng - Đường chéo

Bạn có thể tạo một gradient theo đường chéo bằng cách chỉ định cả vị trí bắt đầu ngang và dọc.

Ví dụ sau đây cho thấy một gradient tuyến tính bắt đầu ở trên cùng bên trái (và đi xuống dưới cùng bên phải). Nó bắt đầu màu đỏ, chuyển sang màu vàng:

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
  height: 200px;
  background-color: red; /* For browsers that do not
support gradients */
  background-image: linear-gradient(to bottom right,
red, yellow);
}
</style>
</head>
<body>

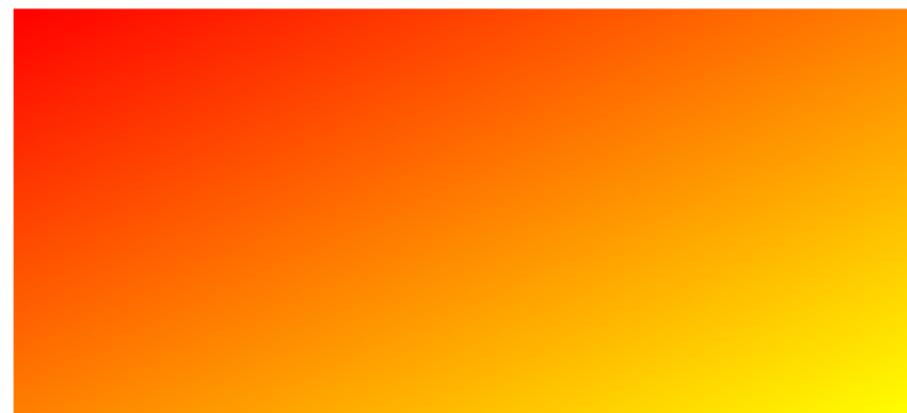
<h1>Linear Gradient - Diagonal</h1>
<p>This linear gradient starts red at top left,
transitioning to yellow (at bottom right):</p>

<div id="grad1"></div>

</body>
</html>
```

Linear Gradient - Diagonal

This linear gradient starts red at top left, transitioning to yellow (at bottom right):



SỬ DỤNG ANGLES

Nếu bạn muốn kiểm soát nhiều hơn hướng của gradient, bạn có thể xác định một góc, thay vì các hướng được xác định trước (xuống dưới, lên trên, sang phải, sang trái, dưới cùng bên phải, v.v.). Giá trị 0deg tương đương với "to top". Giá trị 90deg tương đương với "to right". Giá trị 180deg tương đương với "to bottom".

Cú pháp

```
background-image: linear-gradient(angle, color-stop1, color-stop2);
```

Ví dụ sau cho thấy cách sử dụng các góc trên gradient tuyến tính

```
<html>
<head>
<style>
#grad1 {
    height: 100px;
    background-color: red; /* For browsers that do not support gradients */
    background-image: linear-gradient(0deg, red, yellow);
}

#grad2 {
    height: 100px;
    background-color: red; /* For browsers that do not support gradients */
    background-image: linear-gradient(90deg, red, yellow);
}

#grad3 {
    height: 100px;
    background-color: red; /* For browsers that do not support gradients */
    background-image: linear-gradient(180deg, red, yellow);
}

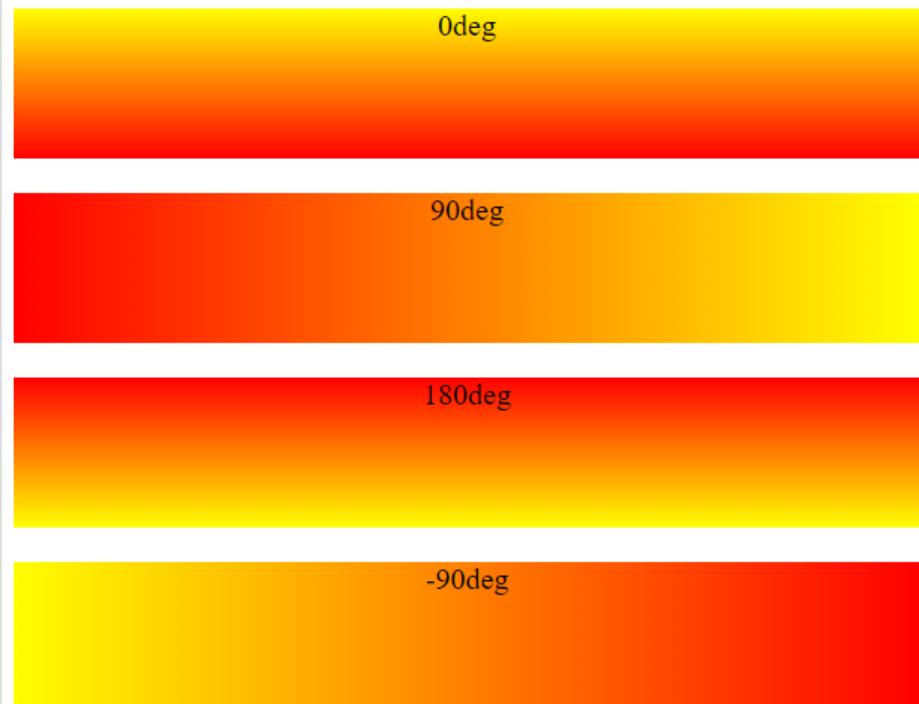
#grad4 {
    height: 100px;
    background-color: red; /* For browsers that do not support gradients */
    background-image: linear-gradient(-90deg, red, yellow);
}
</style>
</head>
<body>

<h1>Linear Gradients - Using Different Angles</h1>

<div id="grad1" style="text-align:center;">0deg</div><br>
<div id="grad2" style="text-align:center;">90deg</div><br>
<div id="grad3" style="text-align:center;">180deg</div><br>
<div id="grad4" style="text-align:center;">-90deg</div>

</body>
</html>
```

Linear Gradients - Using Different Angles



SỬ DỤNG NHIỀU ĐIỂM DỪNG MÀU

Ví dụ sau cho thấy một gradient tuyến tính (từ trên xuống dưới) với nhiều điểm dừng màu:

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 200px;
    background-color: red; /* For browsers that do not support gradients */
    background-image: linear-gradient(red, yellow, green);
}

#grad2 {
    height: 200px;
    background-color: red; /* For browsers that do not support gradients */
    background-image: linear-gradient(red, orange, yellow, green, blue, indigo, violet);
}

#grad3 {
    height: 200px;
```

```
background-color: red; /* For browsers that do not support gradients */
background-image: linear-gradient(red 10%, green 85%, blue 90%);
}
</style>
</head>
<body>

<h1>Linear Gradients - Multiple Color Stops</h1>
<p><strong>Note:</strong> Color stops are spaced evenly when no percents are specified.</p>

<h2>3 Color Stops (evenly spaced):</h2>
<div id="grad1"></div>

<h2>7 Color Stops (evenly spaced):</h2>
<div id="grad2"></div>

<h2>3 Color Stops (not evenly spaced):</h2>
<div id="grad3"></div>

</body>
</html>
```

3 Color Stops (evenly spaced):



7 Color Stops (evenly spaced):



tạo nền cầu vồng

```
#grad {  
    background-image: linear-gradient(to right,  
red,orange,yellow,green,blue,indigo,violet);  
}
```



Nền cầu vồng

SỬ DỤNG TRANSPARENCY

CSS gradient cũng hỗ trợ độ trong suốt, có thể được sử dụng để tạo hiệu ứng mờ dần.

Để thêm độ trong suốt, chúng ta sử dụng hàm `rgba()` để xác định các điểm dừng màu. Tham số cuối cùng trong hàm `rgba()` có thể là một giá trị từ 0 đến 1 và nó xác định độ trong suốt của màu: 0 cho biết độ trong suốt hoàn toàn, 1 cho biết màu đầy đủ (không trong suốt).

Ví dụ sau đây cho thấy một gradient tuyến tính bắt đầu từ bên trái. Nó bắt đầu hoàn toàn trong suốt, chuyển sang màu đỏ đầy đủ:

```
#grad {  
    background-image: linear-gradient(to right, rgba(255,0,0,0), rgba(255,0,0,1));  
}
```



LẬP LẠI LINEAR-GRADIENT



```
#grad {  
  background-image: repeating-linear-gradient(red, yellow 10%, green 20%);  
}
```

CSS GRADIENT

PHẦN 2

CSS RADIAL GRADIENTS

Là Một gradient xuyên tâm được xác định bởi tâm của nó.

Để tạo một gradient xuyên tâm, bạn cũng phải xác định ít nhất hai điểm dừng màu.

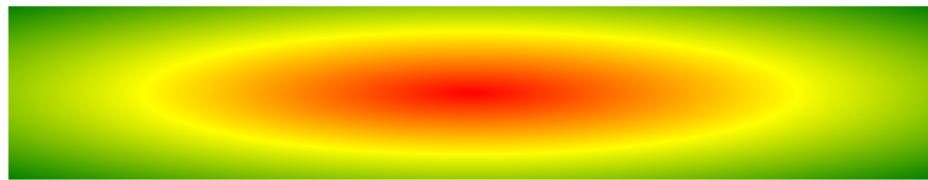
Cú pháp

```
background-image: radial-gradient(shape size at position,  
      start-color, ..., last-color);
```

Theo mặc định, hình dạng là hình elip, kích thước là góc xa nhất và vị trí là trung tâm.

Radial Gradient - Các điểm dừng màu có khoảng cách đều nhau (đây là mặc định)

Ví dụ sau cho thấy một gradient xuyên tâm với các điểm dừng màu cách đều nhau:

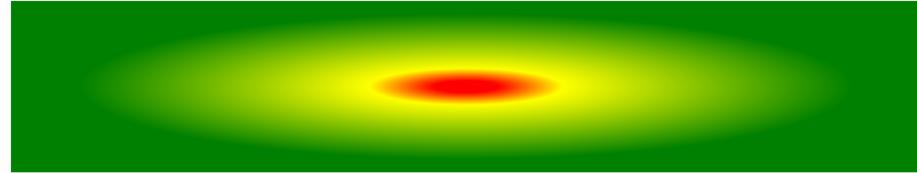


```
#grad {  
    background-image: radial-gradient(red,  
yellow, green);  
}
```

Radial Gradient - Các điểm dừng màu có khoảng cách khác nhau

Ví dụ sau cho thấy một gradient xuyên tâm với các điểm dừng màu cách nhau khác nhau:

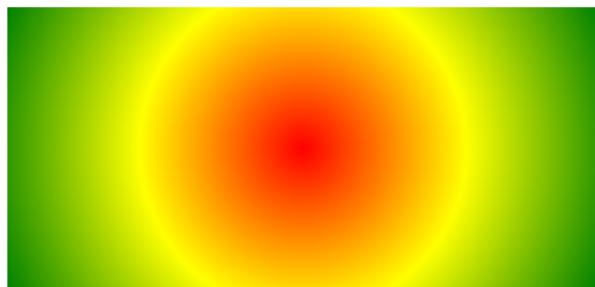
```
#grad {  
    background-image: radial-gradient(red 5%,  
yellow 15%, green 60%);  
}
```



ĐẶT HÌNH DẠNG

Tham số hình dạng xác định hình dạng. Nó có thể lấy giá trị vòng tròn hoặc hình elip. Giá trị mặc định là hình elip.

Ví dụ sau đây cho thấy một gradient xuyên tâm có hình dạng của một vòng tròn:



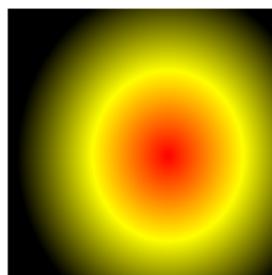
```
#grad {  
    background-image: radial-gradient(circle,  
    red, yellow, green);  
}
```

SỬ DỤNG CÁC TỪ KHÓA CÓ KÍCH THƯỚC KHÁC NHAU

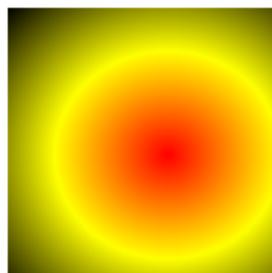
Tham số kích thước xác định kích thước của gradient. Nó có thể nhận bốn giá trị:

- closest-side
- farthest-side
- closest-corner
- farthest-corner

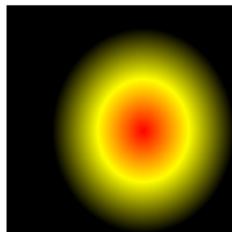
closest-corner:



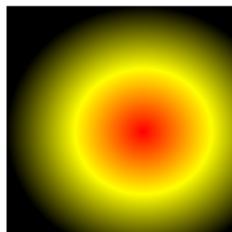
farthest-corner (default):



closest-side:



farthest-side:



```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
    height: 150px;
    width: 150px;
    background-color: red; /* For browsers that
do not support gradients */
    background-image: radial-gradient(closest-
side at 60% 55%, red, yellow, black);
}

#grad2 {
    height: 150px;
    width: 150px;
    background-color: red; /* For browsers that
do not support gradients */
    background-image: radial-gradient(farthest-
side at 60% 55%, red, yellow, black);
}
```

```
#grad3 {  
    height: 150px;  
    width: 150px;  
    background-color: red; /* For browsers that  
do not support gradients */  
    background-image: radial-gradient(closest-  
corner at 60% 55%, red, yellow, black);  
}  
  
#grad4 {  
    height: 150px;  
    width: 150px;  
    background-color: red; /* For browsers that  
do not support gradients */  
    background-image: radial-gradient(farthest-  
corner at 60% 55%, red, yellow, black);  
}  
</style>  
</head>  
<body>  
  
<h1>Radial Gradients – Different size  
keywords</h1>  
  
<h2>closest-side:</h2>  
<div id="grad1"></div>  
  
<h2>farthest-side:</h2>  
<div id="grad2"></div>  
  
<h2>closest-corner:</h2>  
<div id="grad3"></div>  
  
<h2>farthest-corner (default):</h2>  
<div id="grad4"></div>  
  
</body>  
</html>
```



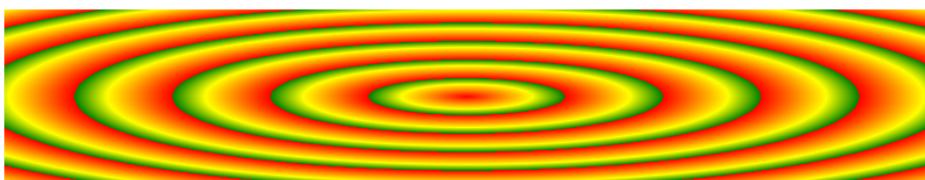
LẶP LẠI MỘT GRADIENT XUYÊN TÂM

Hàm repeat-radial-gradient () được sử dụng để lặp lại các gradient xuyên tâm:

Thí dụ

Một gradient xuyên tâm lặp lại:

```
#grad {  
    background-image: repeating-radial-  
    gradient(red, yellow 10%, green 15%);  
}
```



CSS MULTI COLUMN

CSS MULTI-COLUMN LAYOUT

Bố cục nhiều cột trong CSS cho phép dễ dàng xác định nhiều cột văn bản - giống như trên báo:

Daily Ping

Lorem ipsum
dolor sit amet,
consectetuer adipiscing
elit, sed diam nonummy
nibh euismod tincidunt
ut laoreet dolore magna
aliquam erat volutpat.
Ut wisi enim ad minim
veniam, quis nostrud
exerci tation

ullamcorper suscipit
lobortis nisl ut aliquip ex
ea commodo consequat.
Duis autem vel eum
iriure dolor in hendrerit
in vulputate velit esse
molestie consequat, vel
illum dolore eu feugiat
nulla facilisis at vero
eros et accumsan et
iusto odio dignissim qui

blandit praesent
luptatum zzril delenit
augue duis dolore te
feugait nulla facilisi.
Nam liber tempor cum
soluta nobis eleifend
option congue nihil
imperdiet doming id
quod mazim placerat
facer possim assum.

THUỘC TÍNH NHIỀU CỘT TRONG CSS

Trong chương này, bạn sẽ tìm hiểu về các thuộc tính nhiều cột sau:

- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color
- column-rule
- column-span
- column-width

CSS TẠO NHIỀU CỘT

Các thuộc tính column-count quy định cụ thể số lượng các cột một thẻ html nên được chia thành.

Ví dụ sau sẽ chia văn bản trong phần tử <div> thành 3 cột:

```
<!DOCTYPE html>
```

```
<html>
<head>
<style>
.newspaper {
    column-count: 3;
}
</style>
</head>
<body>

<div class="newspaper">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh
euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim
veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea
commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit
esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et
accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue
duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend
option congue nihil imperdiet doming id quod mazim placerat facer possim assum.
</div>

</body>
</html>
```

Kết quả :

lobortis nisl ut aliquip ex ea
commodo consequat. Duis autem vel
eum iriure dolor in hendrerit in
vulputate velit esse molestie
consequat, vel illum dolore eu
feugiat nulla facilisis at vero eros et
accumsan et iusto odio dignissim qui

blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

CSS CHỈ ĐỊNH KHOẢNG CÁCH GIỮA CÁC CỘT

Các thuộc tính column-gap quy định các khoảng cách giữa các cột.

Ví dụ sau chỉ định khoảng cách 40 pixel giữa các cột:

Thí dụ

```
div {  
    column-gap: 40px;  
}
```

QUY TẮC CỘT CSS

Các thuộc tính column-rule-style xác định phong cách của sự hiển thị giữa các cột:

```
div {  
    column-rule-style: solid;  
}
```

Các thuộc tính column-rule-width quy định độ rộng của sự hiển thị giữa các cột:

```
div {  
    column-rule-width: 1px;  
}
```

Các thuộc tính column-rule-color xác định màu sắc của sự hiển thị giữa các cột:

```
div {  
    column-rule-color: lightblue;  
}
```

Thuộc tính column-rule là cách viết rút gọn của các thuộc tính bắt đầu bằng tiền tố column-rule-*

Ví dụ dưới đây sẽ đặt độ rộng và phong cách , màu sắc của các quy tắc hiển thi giữa các cột

```
div {  
    column-rule: 1px solid lightblue;  
}
```

CHỈ ĐỊNH CHIỀU RỘNG CỘT

Các thuộc tính column-width quy định cụ thể một gợi ý, chiều rộng tối ưu cho các cột.

Ví dụ sau chỉ định rằng chiều rộng được đề xuất, tối ưu cho các cột phải là 100px:

```
div {  
    column-width: 100px;  
}
```

CSS RESIZING

Thuộc tính css resize chỉ định nếu và làm như thế nào 1 thẻ html có thể được thay đổi kích thước bởi người dùng

.

Phần tử div này có thể thay đổi kích thước bởi người dùng!

Để thay đổi kích thước: Nhấp và kéo góc dưới cùng bên phải của phần tử div này.



```
div {  
    resize: horizontal;  
    overflow: auto;  
}
```

Ví dụ sau cho phép người dùng chỉ thay đổi kích thước chiều cao của phần tử <div>:

```
div {  
    resize: vertical;  
    overflow: auto;  
}
```

Ví dụ sau cho phép người dùng thay đổi kích thước cả chiều cao và chiều rộng của phần tử <div>:

```
div {  
    resize: both;  
    overflow: auto;  
}
```

Trong nhiều trình duyệt, thẻ <textarea> được resize là mặc định . Nhưng bạn có thể sử dụng thuộc tính resize với giá trị none để tắt tính năng thay đổi kích thước của thẻ textarea này .

```
textarea {  
    resize: none;  
}
```