

# CSS 值和单位

CSS 中使用的每个属性都有一个值类型，定义了该属性允许的值集。查看 MDN 上的任何属性页面将帮助您了解与对任何特定属性有效的值类型关联的值。在本课中，我们将了解一些最常用的值类型，以及它们最常用的值和单位。

先决条件:	基本的计算机知识、 <a href="#">安装的基本软件、使用文件</a> 的基本知识、HTML 基础知识（学习 <a href="#">HTML 简介</a> ）以及 CSS 的工作原理（学习 <a href="#">CSS 第一步</a> ）。
客观的:	了解 CSS 属性中使用的不同类型的值和单位。

## 什么是 CSS 值？

在 CSS 规范和 MDN 的属性页上，您将能够发现值类型，因为它们将被尖括号包围，例如 [<color>](#) or [<length>](#)。当您看到值类型 `<color>` 对特定属性有效时，这意味着您可以使用任何有效颜色作为该属性的值，如 [<color>](#) 参考页上所列。

**注意：**您还会看到 CSS 值类型称为 *数据类型*。这些术语基本上是可以互换的——当你在 CSS 中看到一些被称为数据类型的东西时，它实际上只是值类型的一种奇特的说法。术语 *值* 是指您选择的值类型支持的任何特定表达式。

**注意：**是的，CSS 值类型倾向于使用尖括号来表示，以将它们与 CSS 属性区分开来（例如，属性 `color` 与 `<color>` 数据类型）。你可能也会混淆 CSS 数据类型和 HTML 元素，因为它们都使用尖括号，但这不太可能——它们在非常不同的上下文中使用。

在下面的示例中，我们使用关键字设置了标题的颜色，并使用函数设置了背景 `rgb()`：

```
h1 {  
  color: black;  
  background-color: rgb(197, 93, 161);  
}
```

CSS 中的值类型是一种定义允许值集合的方法。这意味着，如果您认为 `<color>` 有效，则无需考虑可以使用哪些不同类型的颜色值——关键字、十六进制值、`rgb()` 函数等。您可以使用任何可用 `<color>` 值，前提是您的浏览器支持它们。MDN 上每个值的页面将为您提供有关浏览器支持的信息。例如，如果您查看页面，[<color>](#) 您会看到浏览器兼容性部分列出了不同类型的颜色值和对它们的支持。

让我们看一下您可能经常遇到的一些类型的值和单位，并提供示例以便您可以尝试不同的可能值。

## 数字、长度和百分比

您可能会发现自己在 CSS 中使用了多种数值类型。以下都归类为数字：

数据类型	描述
<a href="#">&lt;integer&gt;</a>	An <code>&lt;integer&gt;</code> 是一个整数，例如 1024 或 -55 。
<a href="#">&lt;number&gt;</a>	A <code>&lt;number&gt;</code> 代表一个十进制数——它可能有也可能没有带小数部分的小数点。例如， 0.255 、 128 或 -1.2 。
<a href="#">&lt;dimension&gt;</a>	A <code>&lt;dimension&gt;</code> 是 <code>&lt;number&gt;</code> 附有一个单位的。例如， 45deg 、 5s 或 10px 。是包括、和 类型 <code>&lt;dimension&gt;</code> 的总括类别 。 <a href="#">&lt;length&gt;</a> <a href="#">&lt;angle&gt;</a> <a href="#">&lt;time&gt;</a> <a href="#">&lt;resolution&gt;</a>
<a href="#">&lt;percentage&gt;</a>	A <code>&lt;percentage&gt;</code> 代表一些其他值的分数。例如， 50% 。百分比值总是相对于另一个数量。例如，元素

数据类型	描述
	的长度是相对于其父元素的长度的。

## 长度

您最常遇到的数字类型是 [<length>](#)。例如， 10px（像素）或 30em。CSS 中使用两种类型的长度——相对长度和绝对长度。重要的是要了解差异，以便了解事情会变得有多大。

### 绝对长度单位

以下都是**绝对**长度单位——它们与其他任何东西都没有关系，通常认为它们的大小总是相同的。

单元	姓名	相当于
cm	厘米	1 厘米 = 37.8 像素 = 25.2/64 英寸
mm	毫米	1mm = 1cm 的 1/10
Q	四分之一毫米	1Q = 1cm 的 1/40
in	英寸	1 英寸 = 2.54 厘米 = 96 像素
pc	毕卡斯	1pc = 1in 的 1/6
pt	积分	1pt = 1in 的 1/72
px	像素	1px = 1in 的 1/96

这些单位中的大多数在用于打印而不是屏幕输出时更有用。例如，我们通常不会 cm 在屏幕上使用（厘米）。您将常用的唯一值是 px（像素）。

### 相对长度单位

相对长度单位是相对于其他东西的，可能是父元素字体的大小，或者视口的大小。使用相对单位的好处是，通过仔细规划，您可以使文本或其他元素的大小相对于页面上的其他所有内容缩放。下表列出了一些对 Web 开发最有用的单元。

单元	关系到
em	父元素的字体大小，在印刷属性的情况下，如 <a href="#">font-size</a> ，以及元素本身的字体大小，在其他属性的情况下，如 <a href="#">width</a> 。
ex	元素字体的 x 高度。
ch	元素字体的字形“O”的提前量度（宽度）。
rem	根元素的字体大小。
lh	元素的行高。
rlh	根元素的行高。当用在根元素的 <a href="#">font-size</a> 或 <a href="#">line-height</a> 属性上时，它指的是属性的初始值。
vw	视口宽度的 1%。
vh	视口高度的 1%。
vmin	视口较小尺寸的 1%。
vmax	视口较大尺寸的 1%。
vb	<a href="#">根元素块轴</a> 方向初始包含块大小的 1%。
vi	<a href="#">根元素内联轴</a> 方向初始包含块大小的 1%。
svw, svh	<a href="#">分别为小视口</a> 宽度和高度的 1%。
lvw, lvh	<a href="#">分别为大视口</a> 宽度和高度的 1%。
dvw, dvh	<a href="#">分别为动态视口</a> 宽度和高度的 1%。

### 探索一个例子

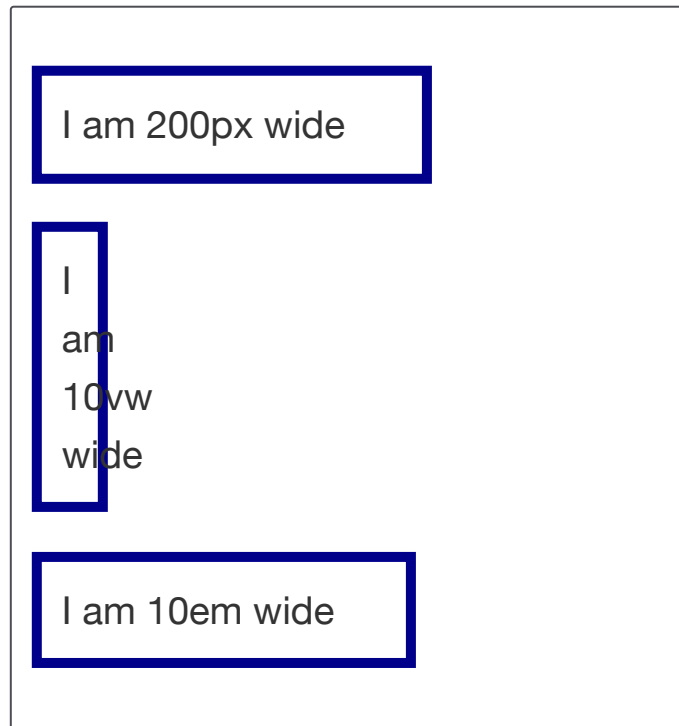
在下面的示例中，您可以看到一些相对和绝对长度单位的行为。第一个框有一 [width](#) 组像素。作为一个绝对单位，这个宽度无论发生什么变化都将保

持不变。

第二个框的宽度设置为 `vw`（视口宽度）单位。该值是相对于视口宽度的，因此 `10vw` 是视口宽度的 10%。如果您更改浏览器窗口的宽度，则框的大小也应更改。但是，此示例是使用嵌入到页面中的 `<iframe>`，因此这将不起作用。要实际查看此示例，您必须[在其自己的浏览器选项卡中打开该示例后尝试该示例](#)。

第三个框使用 `em` 单位。这些是相对于字体大小的。1em 我在包含的字体大小上设置了 `<div>`，它有一个类 `.wrapper`。将此值更改为 `1.5em`，您会看到所有元素的字体大小都增加了，但只有最后一项会变宽，因为它的宽度是相对于该字体大小的。

按照上述说明进行操作后，尝试以其他方式使用这些值，看看会得到什么。



## Interactive editor

```
.wrapper {  
  font-size: 1em;  
}  
  
.px {  
  width: 200px;  
}  
  
.vw {  
  width: 10vw;  
}  
  
.em {  
  width: 10em;  
}
```

```
<div class="wrapper">  
  <div class="box px">I am 200px  
  wide</div>  
  <div class="box vw">I am 10vw  
  wide</div>  
  <div class="box em">I am 10em  
  wide</div>  
</div>
```

Reset

## ems 和 rems

em 并且 rem 是您在调整从框到文本的任何内容时最常遇到的两个相对长度。了解它们是如何工作的，以及它们之间的区别是值得的，尤其是当您开始接触更复杂的主题（如样式[文本](#)或[CSS 布局](#)）时。下面的例子提供了一个演示。

下面所示的 HTML 是一组嵌套列表——我们总共有三个列表，两个示例都有相同的 HTML。唯一的区别是第一个有 *ems* 类，第二个有 *rems* 类。

首先，我们将 16px 设置为 `<html>` 元素的字体大小。

回顾一下，在排版的情况下，**em 单位**意味着“我的父元素的字体大小”。with a of `<li>` 中的元素从它们的父元素中获取它们的大小。因此，每个连续的嵌套级别都会逐渐变大，因为每个嵌套的字体大小都设置为其父级字体大小的 1.3 倍。`<ul> class ems 1.3em`

概括地说，**rem 单位**表示“根元素的字体大小”（rem 代表“root em”）。`<li>` 带有 of 的元素 `<ul>` 从 class 根元素 `rems ( <html> )` 获取它们的大小。这意味着每个连续的嵌套级别不会不断变大。

但是，如果您更改 CSS 中的 `<html>` 元素，您将看到其他所有内容都相对于它发生了变化——包括 - 和 - 大小的文本。font-size rem em

- One
- Two
- Three
  - Three A
  - Three B
    - Three B 2

- One
- Two
- Three
  - Three A
  - Three B
    - Three B 2

### Interactive editor

```
html {  
  font-size: 16px;  
}  
  
.ems li {  
  font-size: 1.3em;  
}  
  
.rem li {  
  font-size: 1.3rem;  
}
```

```
<ul class="ems">  
  <li>One</li>  
  <li>Two</li>  
  <li>Three  
    <ul>  
      <li>Three A</li>
```



## 百分比

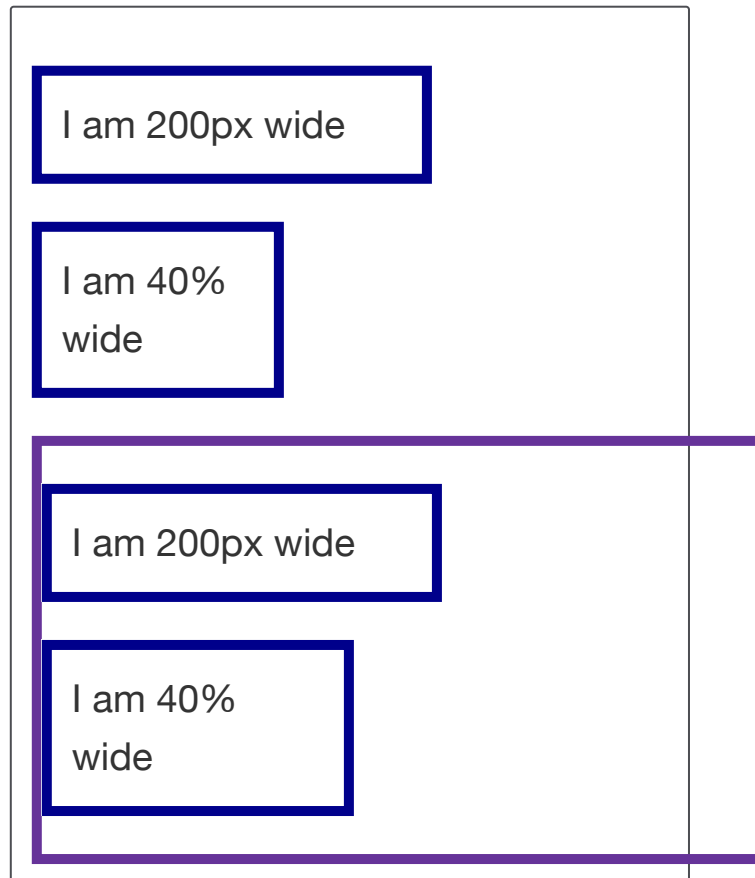
[Reset](#)

在很多情况下，百分比的处理方式与长度相同。百分比的问题在于它们总是相对于其他一些值设置的。例如，如果您将一个元素设置为百分比，它将是该元素父元素的 `font-size` 百分比。`font-size` 如果您使用百分比作为值，它将是父项的 `width` 百分比。`width`

在下面的示例中，两个百分比大小的框和两个像素大小的框具有相同的类名。这些集合分别为 40% 和 200px 宽。

不同之处在于第二组两个框位于 400 像素宽的包装器内。第二个 200px 宽的框与第一个宽度相同，但第二个 40% 的框现在是 400px 的 40%——比第一个窄很多！

尝试更改包装器的宽度或百分比值以查看其工作原理。



## Interactive editor

```
.wrapper {  
  width: 400px;  
  border: 5px solid rebeccapurple;  
}  
  
.px {  
  width: 200px;  
}  
  
.percent {  
  width: 40%;  
}
```

```
<div class="box px">I am 200px  
wide</div>  
<div class="box percent">I am 40%  
wide</div>  
<div class="wrapper">  
  <div class="box px">I am 200px  
wide</div>  
  <div class="box percent">I am
```

Reset

下一个示例以百分比设置字体大小。每个 `<li>` 都有 `font-size 80%`；因此，嵌套的列表项会随着从父级继承大小而逐渐变小。

- One
- Two
- Three
  - Three A
  - Three B
    - Three B 2

### Interactive editor

```
li {  
  font-size: 80%;  
}
```

```
<ul>  
  <li>One</li>  
  <li>Two</li>  
  <li>Three  
    <ul>  
      <li>Three A</li>  
      <li>Three B  
        <ul>  
          <li>Three B 2</li>  
        </ul>  
      </li>  
    </ul>  
  </li>  
</ul>
```

Reset

请注意，虽然许多值类型接受长度或百分比，但有些只接受长度。您可以在 MDN 属性参考页面上查看接受哪些值。如果允许的值包括，[<length-](#)

[percentage](#)> 那么您可以使用长度或百分比。如果允许的值只包括 `<length>`，则不能使用百分比。

## 数字

一些值类型接受数字，不添加任何单位。接受无单位数字的属性的一个例子是属性 `opacity`，它控制元素的不透明度（它的透明度）。0 此属性接受介于（完全透明）和（完全不透明）之间的数字 1。

在下面的示例中，尝试将 的值更改为和 `opacity` 之间的各种十进制值，并查看框及其内容如何变得或多或少不透明。 0 1

I am a box with  
opacity

### Interactive editor

```
.box {  
  opacity: 0.6;  
}
```

```
<div class="wrapper">  
  <div class="box">I am a box with  
  opacity</div>  
</div>
```

Reset

**注意：**当您在 CSS 中使用数字作为值时，不应将其括在引号中。

## 颜色

在 CSS 中指定颜色的方法有很多，其中一些比其他方法最近才实现。相同的颜色值可以在 CSS 中的任何地方使用，无论您是指定文本颜色、背景颜色还是其他任何颜色。

现代计算机中可用的标准颜色系统支持 24 位颜色，它允许通过不同的红色、绿色和蓝色通道的组合显示大约 1670 万种不同的颜色，每个通道有 256 个不同的值（ $256 \times 256 \times 256 = 16,777,216$ ）。让我们看一下在 CSS 中指定颜色的一些方法。

**注意：**在本教程中，我们将研究具有良好浏览器支持的指定颜色的常用方法；还有其他的，但他们没有那么好的支持，而且不太常见。

## 颜色关键词

在学习部分或 MDN 其他地方的示例中，您经常会看到使用的颜色关键字，因为它们是指定颜色的一种简单易懂的方式。这些关键字有很多，其中一些具有相当有趣的名称！您可以在页面上看到值类型的完整列表 [<color>](#)。

在下面的实例中尝试使用不同的颜色值，以更多地了解它们的工作原理。

antiquewhite

blueviolet

greenyellow

## Interactive editor

```
.one {  
  background-color: antiquewhite;  
}  
  
.two {  
  background-color: blueviolet;  
}  
  
.three {  
  background-color: greenyellow;  
}
```

```
<div class="wrapper">  
  <div class="box  
one">antiquewhite</div>  
  <div class="box  
two">blueviolet</div>  
  <div class="box  
three">greenyellow</div>  
</div>
```



Reset

## 十六进制 RGB 值

您可能会遇到的下一种颜色值是十六进制代码。每个十六进制值都包含一个井号/井号 (#)，后跟六个十六进制数，每个十六进制数都可以取 0 到 f（代表 15）之间的 16 个值之一——所以 0123456789abcdef。每对值代表一个

通道——红色、绿色和蓝色——并允许我们为每个通道指定 256 个可用值中的任何一个 ( $16 \times 16 = 256$ )。

这些值有点复杂，也不太容易理解，但它们比关键字更通用——您可以使用十六进制值来表示要在配色方案中使用的任何颜色。

#02798b

#c55da1

#128a7d

### Interactive editor

```
.one {  
  background-color: #02798b;  
}  
  
.two {  
  background-color: #c55da1;  
}  
  
.three {  
  background-color: #128a7d;  
}
```

```
<div class="wrapper">  
  <div class="box one">#02798b</div>  
  <div class="box two">#c55da1</div>  
  <div class="box  
three">#128a7d</div>  
</div>
```

Reset

再次尝试更改值以查看颜色如何变化。

## RGB 和 RGBA 值

我们要在这里讨论的第三种方案是 RGB。RGB 值是一个函数——`rgb()` 它被赋予三个参数，分别代表颜色的红色、绿色和蓝色通道值，与十六进制值的方式非常相似。与 RGB 的不同之处在于，每个通道不是由两个十六进制数字表示，而是由 0 到 255 之间的十进制数字表示——更容易理解。

让我们重写上一个示例以使用 RGB 颜色：



rgb(2, 121, 139)

rgb(197, 93, 161)

rgb(18, 138, 125)

## Interactive editor

```
.one {  
  background-color: rgb(2, 121, 139);  
}  
  
.two {  
  background-color: rgb(197, 93,  
161);  
}  
  
.three {  
  background-color: rgb(18, 138,  
125);  
}
```

```
<div class="wrapper">  
  <div class="box one">rgb(2,  
121, 139)</div>  
  <div class="box two">rgb(197,  
93, 161)</div>  
  <div class="box three">rgb(18,  
138, 125)</div>  
</div>
```



Reset

您可以将第四个参数传递给 `rgb()`，它表示控制不透明度的颜色的 alpha 通道。如果将此值设置为 `0` 它将使颜色完全透明，反之则 `1` 使其完全不透明。介于两者之间的值可为您提供不同级别的透明度。

**注意：**在颜色上设置 alpha 通道与使用 [opacity](#) 我们之前看到的属性有一个关键区别。当您使用不透明度时，您会使元素及其内

部的所有内容不透明，而使用带有 alpha 参数颜色的 RGB 只会使您指定的颜色不透明。

在下面的示例中，我们已将背景图像添加到彩色框的包含块中。然后我们将框设置为具有不同的不透明度值——注意当 alpha 通道值较小时背景如何显示得更多。

rgba(2, 121, 139, .3)

rgba(197, 93, 161, .7)

rgba(18, 138, 125, .9)

## Interactive editor

```
.one {  
  background-color: rgba(2, 121,  
139, .3);  
}  
  
.two {  
  background-color: rgba(197, 93,  
161, .7);  
}  
  
.three {  
  background-color: rgba(18, 138,  
125, .9);  
}
```

```
<div class="wrapper">  
  <div class="box one">rgba(2,  
121, 139, .3)</div>  
  <div class="box two">rgba(197,  
93, 161, .7)</div>  
  <div class="box three">rgba(18,  
138, 125, .9)</div>  
</div>
```

Reset

在此示例中，尝试更改 alpha 通道值以查看它如何影响颜色输出。

**注意：**在旧版本的 CSS 中，`rgb()` 语法不支持 `alpha` 参数——你需要使用一个不同的函数来调用 `rgba()` 它。如今，您可以将 `alpha` 参数传递给 `rgb()`，但为了与旧网站向后兼容，`rgba()` 语法仍然受支持，并且具有与 `rgb()` 。

## HSL 和 HSLA 值

另一种指定颜色的方法是 HSL 颜色模型。该函数不接受红色、绿色和蓝色值，而是 `hsl()` 接受色调、饱和度和亮度值，这些值用于区分 1670 万种颜色，但方式不同：

- **色调：**颜色的基础色调。[这采用 0 到 360 之间的值，表示围绕色轮的角度。](#)
- **饱和度：**颜色的饱和度如何？这取一个 0-100% 的值，其中 0 是无颜色（它将显示为灰色阴影），100% 是全色饱和度
- **亮度：**颜色有多亮或多亮？这取一个 0-100% 的值，其中 0 是没有光（它会呈现全黑），100% 是全光（它会呈现全白）

我们可以像这样更新 RGB 示例以使用 HSL 颜色：

hsl(188, 97%, 28%)

hsl(321, 47%, 57%)

hsl(174, 77%, 31%)

## Interactive editor

```
.one {  
  background-color: hsl(188, 97%,  
28%);  
}  
  
.two {  
  background-color: hsl(321, 47%,  
57%);  
}  
  
.three {  
  background-color: hsl(174, 77%,  
31%);  
}
```

```
<div class="wrapper">  
  <div class="box one">hsl(188,  
97%, 28%)</div>  
  <div class="box two">hsl(321,  
47%, 57%)</div>  
  <div class="box three">hsl(174,  
77%, 31%)</div>  
</div>
```

Reset

就像 `rgb()` 你可以传递一个 `alpha` 参数来 `hsl()` 指定不透明度：

hsl(188 97% 28% / .3)

hsl(321 47% 57% / .7)

hsl(174 77% 31% / .9)

## Interactive editor

```
.one {  
  background-color: hsl(188 97%  
28% / .3);  
}  
  
.two {  
  background-color: hsl(321 47%  
57% / .7);  
}
```



在这个领先的平台上  
构建、训练、测试和  
部署智能虚拟助手  
——免费试用。

[Mozilla 广告](#)

不想看广告？

```
}  
  
<div class="wrapper">  
  <div class="box one">hsl(188  
97% 28% / .3)</div>  
  <div class="box two">hsl(321  
47% 57% / .7)</div>  
  <div class="box three">hsl(174  
77% 31% / .9)</div>  
</div>
```



Reset

**注意：**在旧版本的 CSS 中，`hsl()` 语法不支持 alpha 参数——你需要使用一个不同的函数来调用 `hsla()` 它。如今，您可以将

alpha 参数传递给 `hsl()`，但为了与旧网站向后兼容，`hsla()` 语法仍然受支持，并且具有与 `hsl()` 。

您可以在项目中使用任何这些颜色值。对于大多数项目，您可能会决定一个调色板，然后在整个项目中使用这些颜色——以及您选择的指定颜色的方法。您可以混合和匹配颜色模型，但为了保持一致性，通常最好整个项目都使用相同的模型！

## 图片

[`<image>`](#) 只要图像是有效值，就会使用值类型。`url()` 这可以通过函数或渐变指向的实际图像文件。

在下面的示例中，我们演示了用作 CSS 属性值的图像和渐变 `background-image` 。



Interactive editor

```
.image {
  background-image: url(star.png);
}

.gradient {
  background-image: linear-gradient(90deg,
    rgba(119,0,255,1) 39%, rgba(0,212,255,1) 100%);
}
```

```
<div class="box image"></div>
<div class="box gradient"></div>
```

Reset

注意：还有一些其他可能的值 `<image>`，但是这些值较新并且当前浏览器支持较差。 `<image>` 如果您想阅读有关数据类型的信



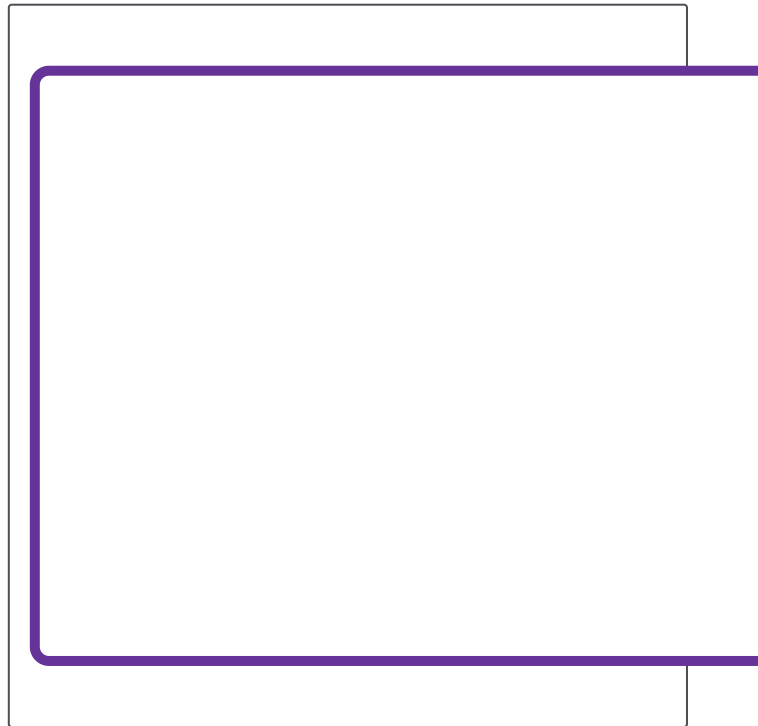
息，请查看 MDN 上的页面。

## 位置

值 `<position>` 类型表示一组 2D 坐标，用于定位项目，例如背景图像（通过 `background-position`）。它可以采用诸如 `top`，`left`，`bottom`，`right`，之类的关键字，并将 `center` 项目与 2D 框的特定边界对齐，以及表示距框的顶部和左侧边缘的偏移量的长度。

典型的位置值由两个值组成——第一个设置水平位置，第二个垂直设置位置。如果您只为一个轴指定值，另一个将默认为 `center`。

在下面的示例中，我们使用关键字将背景图像定位在距离容器顶部和右侧 40 像素的位置。



## Interactive editor

```
.box {  
  height: 300px;  
  width: 400px;  
  background-image: url(star.png);  
  background-repeat: no-repeat;  
  background-position: right 40px;  
}
```



```
<div class="box"></div>
```



Reset

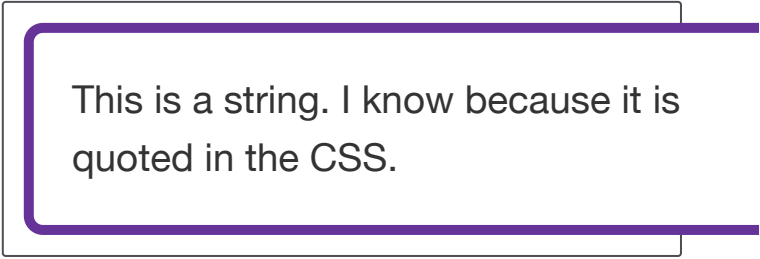
尝试使用这些值，看看如何推动图像。

## 字符串和标识符

在上面的示例中，我们看到了关键字用作值的地方（例如 `<color>` 关键字，如 `red`、`black`、`rebeccapurple` 和 `goldenrod`）。这些关键字更准

确地描述为 *identifiers*，CSS 理解的特殊值。因此它们不被引用——它们不被视为字符串。

在 CSS 中有些地方使用字符串。例如，[指定生成的内容时](#)。在这种情况下，引用该值以表明它是一个字符串。在下面的示例中，我们使用不带引号的颜色关键字以及带引号的生成内容字符串。



This is a string. I know because it is quoted in the CSS.

### Interactive editor

```
.box {  
  width:400px;  
  padding: 1em;  
  border-radius: .5em;  
  border: 5px solid rebeccapurple;  
  background-color: lightblue;  
}  
  
.box::after {  
  content: "This is a string. I know  
  because it is quoted in the CSS."  
}
```

```
<div class="box"></div>
```

Reset

## 功能

我们要看的最后一种值是称为函数的值组。在编程中，函数是一段可重复使用的代码，可以多次运行以完成重复性任务，而开发人员和计算机都需要付出最少的努力。函数通常与 JavaScript、Python 或 C++ 等语言相关联，但它们也作为属性值存在于 CSS 中。我们已经在“颜色”部分看到了函数的

作用—— `rgb()`、`hsl()` 等。用于从文件返回图像的值 `url()` ——也是一个函数。

CSS 函数是一个行为更像您在传统编程语言中可能会发现的东西的值 `calc()`。此函数使您能够在 CSS 中进行简单的计算。如果您想计算出您在为项目编写 CSS 时无法定义的值，并且需要浏览器在运行时为您计算出值，那么它特别有用。

例如，下面我们用来 `calc()` 使框 `20% + 100px` 变宽。`20%` 是根据父容器的宽度计算得出的 `.wrapper`，因此如果该宽度发生变化，也会发生变化。我们无法事先进行此计算，因为我们不知道父级的 `20%` 是多少，所以我们使用 `calc()` 告诉浏览器为我们做这件事。

My width is calculated.

### Interactive editor

```
.wrapper {  
  width: 400px;  
}  
  
.box {  
  width: calc(20% + 100px);  
}
```

```
<div class="wrapper">  
  <div class="box">My width is  
  calculated.</div>  
</div>
```

Reset

## 测试你的技能！

您已读完本文，但您还记得最重要的信息吗？在继续之前，您可以找到一些进一步的测试来验证您是否记住了这些信息——请参阅[测试您的技能：值和单位](#)。

## 概括

这是对您可能遇到的最常见类型的值和单位的快速浏览。[您可以在CSS 值和单位](#)参考页面上查看所有不同的类型——在学习这些课程时，您会遇到许多正在使用的类型。

要记住的关键是每个属性都有一个定义的允许值类型列表，并且每个值类型都有一个解释值是什么的定义。然后你可以在 MDN 上查找细节。例如，了解 [<image>](#) 还允许您创建颜色渐变是有用的，但可能不是显而易见的知识！

在下一篇文章中，我们将了解如何在 CSS 中[调整项目的大小](#)。

此页面最后修改于 2023 年 2 月 23 日由[MDN 贡献者](#)提供。