CSS 的结构

现在您开始了解 CSS 的目的和用途，让我们检查一下 CSS 的结构。

先决条件：	基本的计算机知识、 安装的基本软件、使用文件的基本知识、HTML 基础知识（学习 HTML 简介）以及CSS 工作原理的概念 。
客观的：	详细学习CSS的基本语法结构。

将 CSS 应用到 HTML

首先，我们来看看将 CSS 应用于文档的三种方法：使用外部样式表、使用内部样式表和使用内联样式。

外部样式表

外部样式表在带有 `.css` 扩展名的单独文件中包含 CSS。这是将 CSS 引入文档的最常见和最有用的方法。您可以将单个 CSS 文件链接到多个网页，并使用相同的 CSS 样式表对所有网页进行样式设置。在CSS 入门中，我们将外部样式表链接到我们的网页。

从 HTML 元素引用外部 CSS 样式表 `<link>`：

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <meta charset="utf-8" />
    <title>My CSS experiment</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <h1>Hello World!</h1>
```

```
    <p>This is my first CSS example</p>
  </body>
</html>
```

CSS 样式表文件可能如下所示：

```
h1 {
  color: blue;
  background-color: yellow;
  border: 1px solid black;
}

p {
  color: red;
}
```

`href` 元素的属性需要 `<link>` 引用文件系统上的文件。在上面的示例中，CSS 文件与 HTML 文档位于同一文件夹中，但您可以将其放在其他位置并调整路径。以下是三个示例：

```
<!-- Inside a subdirectory called styles inside the current
directory -->
<link rel="stylesheet" href="styles/style.css" />

<!-- Inside a subdirectory called general, which is in a
subdirectory called styles, inside the current directory -->
<link rel="stylesheet" href="styles/general/style.css" />

<!-- Go up one directory level, then inside a subdirectory
called styles -->
<link rel="stylesheet" href="../styles/style.css" />
```

内部样式表

内部样式表驻留在 HTML 文档中。要创建内部样式表，您可以将 CSS 放在 `<style>` HTML 中包含的元素中 `<head>` 。

内部样式表的 HTML 可能如下所示：

```
<!DOCTYPE html>
<html lang="en-GB">
```

```
  <head>
    <meta charset="utf-8" />
    <title>My CSS experiment</title>
    <style>
      h1 {
        color: blue;
        background-color: yellow;
        border: 1px solid black;
      }

      p {
        color: red;
      }
    </style>
  </head>
  <body>
    <h1>Hello World!</h1>
    <p>This is my first CSS example</p>
  </body>
</html>
```

在某些情况下，内部样式表可能很有用。例如，也许您正在使用一个内容管理系统，在该系统中您被禁止修改外部 CSS 文件。

但是对于具有多个页面的站点，内部样式表成为一种效率较低的工作方式。要使用内部样式表将统一的 CSS 样式应用于多个页面，您必须在每个将使用该样式的网页中都有一个内部样式表。效率损失也会影响站点维护。使用内部样式表中的 CSS，存在这样的风险，即即使是一个简单的样式更改也可能需要对多个网页进行编辑。

行内样式

内联样式是影响包含在 `style` 属性中的单个 HTML 元素的 CSS 声明。HTML 文档中内联样式的实现可能如下所示：

```
<!DOCTYPE html>
<html lang="en-GB">
  <head>
    <meta charset="utf-8" />
    <title>My CSS experiment</title>
  </head>
  <body>
```

```
    <h1 style="color: blue;background-color: yellow;border: 1px
solid black;">
      Hello World!
    </h1>
    <p style="color:red;">This is my first CSS example</p>
  </body>
</html>
```

尽可能避免以这种方式使用 CSS。这与最佳实践相反。首先，它是用于维护的 CSS 的最低效实现。一次样式更改可能需要在单个网页中进行多次编辑。其次，内联 CSS 还将 (CSS) 表示代码与 HTML 和内容混合在一起，使所有内容都更加难以阅读和理解。分离代码和内容使所有在网站上工作的人都更容易维护。

在某些情况下，内联样式更为常见。如果您的工作环境非常受限，您可能不得不求助于使用内联样式。例如，您的 CMS 可能只允许您编辑 HTML 正文。您可能还会在 HTML 电子邮件中看到很多内联样式，以实现与尽可能多的电子邮件客户端的兼容性。

玩转本文中的 CSS

对于接下来的练习，请在您的计算机上创建一个文件夹。您可以随意命名该文件夹。在文件夹中，复制以下文本以创建两个文件：

索引.html：

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width" />
    <title>My CSS experiments</title>
    <link rel="stylesheet" href="styles.css" />
  </head>
  <body>
    <p>Create your test HTML here</p>
  </body>
</html>
```

样式.css：

```
/* Create your test CSS here */

p {
  color: red;
}
```

当您找到想要试验的 CSS 时，将 HTML `<body>` 内容替换为一些要设置样式的 HTML，然后将您的测试 CSS 代码添加到您的 CSS 文件中。

或者，您也可以使用下面的交互式编辑器。

What color am I?

Interactive editor

```
/* Create your test CSS here */

.special {
  color: red;
}

p {
  color: blue;
}
```

```
<p class="special">What color am I?</p>
```

Reset

继续阅读，玩得开心！

选择器

选择器以 HTML 为目标，将样式应用于内容。我们已经在CSS 入门教程中发现了多种选择器。如果 CSS 没有按预期应用于内容，您的选择器可能不会按照您认为应该匹配的方式匹配。

每个 CSS 规则都以一个选择器（或选择器列表）开头，以便告诉浏览器规则应该应用于哪个或哪些元素。下面的所有示例都是有效的选择器或选择器列表。

```
h1
a:link
.manythings
#onething
*
.box p
.box p:first-child
h1, h2, .intro
```

尝试创建一些使用上述选择器的 CSS 规则。添加由选择器设置样式的 HTML。如果上面的任何语法不熟悉，请尝试搜索 MDN。

> **注意：**您将在下一模块中了解有关选择器的更多信息：CSS 选择器。

特异性

你可能会遇到两个选择器选择同一个 HTML 元素的场景。考虑下面的样式表，其中一个 `p` 选择器将段落文本设置为蓝色。但是，还有一个类可以将所选元素的文本设置为红色。

```
.special {
  color: red;
}

p {
  color: blue;
}
```

假设在我们的 HTML 文档中，我们有一个类为 `special` . 两条规则都适用。哪个选择器占优势？您希望看到蓝色或红色段落文本吗？

```
<p class="special">What color am I?</p>
```

CSS 语言有规则来控制在发生冲突时哪个选择器更强。这些规则称为**级联**和**特异性**。在下面的代码块中，我们为 p 选择器定义了两个规则，但段落文本将为蓝色。这是因为将段落文本设置为蓝色的声明稍后出现在样式表中。后面的样式会替换样式表中较早出现的冲突样式。这就是**级联**规则。

```
p {
  color: red;
}

p {
  color: blue;
}
```

然而，在我们之前的示例中，类选择器和元素选择器之间存在冲突，类优先，将段落文本渲染为红色。即使稍后在样式表中出现了冲突的样式，这怎么会发生呢？一个类被评为更具体，因为它比元素选择器更**具体，因此它取消了其他冲突的样式声明。**

亲自尝试这个实验！添加 HTML，然后将这两个 `p { }` 规则添加到您的样式表中。接下来，将第一个 p 选择器更改为 `.special` 以查看它如何更改样式。

特异性和级联的规则起初看起来很复杂。当您对 CSS 越来越熟悉时，这些规则会更容易理解。下一模块中的级联和继承部分对此进行了详细解释，包括如何计算特异性。

现在，请记住存在特异性。有时，CSS 可能不会像您预期的那样应用，因为样式表中的其他内容具有更多的特异性。认识到多个规则可以应用于一个元素是解决此类问题的第一步。

属性和值

在最基本的层面上，CSS 由两个组件组成：

- **属性**：这些是人类可读的标识符，指示您要修改的样式特征。例如，`font-size` ， `width` ， `background-color` 。
- **值**：每个属性都分配了一个值。此值指示如何设置属性的样式。

下面的示例突出显示了单个属性和值。属性名称为 `color` ， 值为 `blue` 。

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

当属性与值配对时，这种配对称为*CSS 声明*。CSS 声明位于*CSS 声明块*中。在下面的示例中，突出显示标识 CSS 声明块。

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

*最后，CSS 声明块与选择器*配对以生成*CSS 规则集*（或*CSS 规则*）。下面的示例包含两个规则：一个用于 `h1` 选择器，一个用于 `p` 选择器。彩色突出显示标识 `h1` 规则。

```
h1 {
  color: blue;
  background-color: yellow;
}

p {
  color: red;
}
```

将 CSS 属性设置为特定值是定义文档布局和样式的主要方式。CSS 引擎计算哪些声明适用于页面的每个元素。

CSS 属性和值不区分大小写。属性值对中的属性和值由冒号 ( ) 分隔 ： 。

查找下面列出的属性的不同值。编写将样式应用于不同 HTML 元素的 CSS 规则：

- `font-size`
- `width`
- `background-color`
- `color`
- `border`

> **警告：** 如果属性未知，或者值对于给定属性无效，则声明将被处理为*无效*。浏览器的 CSS 引擎完全忽略它。

> **警告：** 在 CSS（和其他 Web 标准）中，人们一致认为美国拼写是存在语言变化或不确定性的标准。例如， `colour` 应该拼写 `color` 为 as `colour` 将不起作用。

功能

虽然大多数值是相对简单的关键字或数值，但也有一些值采用函数的形式。

计算（）函数

一个例子是 `calc()` 函数，它可以在 CSS 中做简单的数学运算：

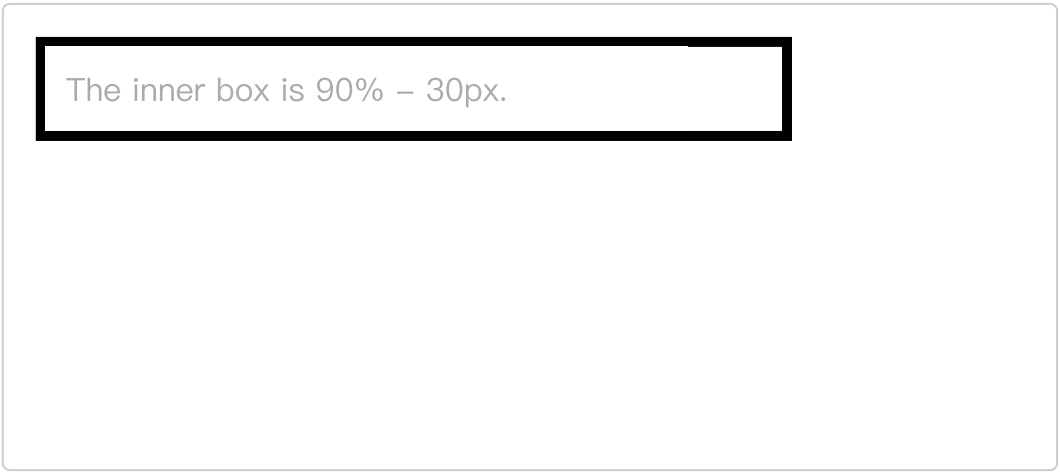
```
<div class="outer"><div class="box">The inner box is 90% - 30px.
</div></div>
```

```
.outer {
  border: 5px solid black;
```

```
}

.box {
  padding: 10px;
  width: calc(90% - 30px);
  background-color: rebeccapurple;
  color: white;
}
```

这呈现为：



函数由函数名和括起函数值的圆括号组成。在上面的示例中 `calc()`，这些值将此框的宽度定义为包含块宽度的 90%，减去 30 像素。计算结果不是可以预先计算并作为静态值输入的结果。

变换函数

另一个例子是 的各种值 `transform`，例如 `rotate()`。

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

```
.box {
  margin: 30px;
  width: 100px;
  height: 100px;
  background-color: rebeccapurple;
  transform: rotate(0.8turn);
}
```

上述代码的输出如下所示：

查找下面列出的属性的不同值。编写将样式应用于不同 HTML 元素的 CSS 规则：

- `transform`
- `background-image`，**特别是梯度值**
- `color`，**特别是 rgb 和 hsl 值**

@规则

CSS @rules（发音为“at-rules”）提供了 CSS 应该执行什么或应该如何运行的指令。有些@rules 很简单，只有一个关键字和一个值。例如，`@import` 将样式表导入另一个 CSS 样式表：

```
@import "styles2.css";
```

您可能会遇到的一种常见@rule 是 `@media`，它用于创建媒体查询。媒体查询使用条件逻辑来应用 CSS 样式。

在下面的示例中，样式表为元素定义了默认的粉红色背景 `<body>`。但是，如果浏览器视口宽度超过 30em，则随后的媒体查询会定义蓝色背景。

```
body {
  background-color: pink;
}

@media (min-width: 30em) {
  body {
    background-color: blue;
```

```
  }
}
```

在这些教程中，您将遇到其他@rules。

看看您是否可以添加一个根据视口宽度更改样式的媒体查询。更改浏览器窗口的宽度以查看结果。

速记

font 、 background 、 padding 、 border 和 等属性 margin 称为**速记属性**。这是因为速记属性在一行中设置了多个值。

例如，这一行代码：

```
/* In 4-value shorthands like padding and margin, the values are
applied
   in the order top, right, bottom, left (clockwise from the
top). There are also other
   shorthand types, for example 2-value shorthands, which set
padding/margin
   for top/bottom, then left/right */
padding: 10px 15px 15px 5px;
```

相当于这四行代码：

```
padding-top: 10px;
padding-right: 15px;
padding-bottom: 15px;
padding-left: 5px;
```

这一行：

```
background: red url(bg-graphic.png) 10px 10px repeat-x fixed;
```

相当于这五行：

```
background-color: red;
background-image: url(bg-graphic.png);
background-position: 10px 10px;
background-repeat: repeat-x;
background-attachment: fixed;
```

在本课程的后面，您将遇到许多其他速记属性示例。MDN 的CSS 参考是获取有关任何速记属性的更多信息的良好资源。

尝试在您自己的 CSS 练习中使用声明（以上），以更加熟悉它的工作原理。您还可以尝试不同的值。

> **警告：**使用 CSS 速记的一个不太明显的方面是如何重置省略的值。未在 CSS 速记中指定的值恢复为初始值。这意味着 CSS 速记中的遗漏可以**覆盖以前设置的值**。

评论

与任何编码工作一样，最佳做法是将注释与 CSS 一起编写。这有助于您在稍后返回进行修复或增强时记住代码的工作方式。它还可以帮助其他人理解代码。

CSS 注释以 开头 `/*` 和结尾 `*/` 。在下面的示例中，注释标记不同代码段的开始。这有助于在代码库变大时导航代码库。有了这种注释，在代码编辑器中搜索注释就变成了一种高效查找代码段的方法。

```
/* Handle basic element styling */
/* ------------------------------------------------------------
------------------------------ */
body {
  font: 1em/150% Helvetica, Arial, sans-serif;
  padding: 1em;
  margin: 0 auto;
  max-width: 33em;
}

@media (min-width: 70em) {
```

```
  /* Increase the global font size on larger screens or windows
     for better readability */
  body {
    font-size: 130%;
  }
}

h1 {
  font-size: 1.5em;
}

/* Handle specific elements nested in the DOM */
div p,
#id:first-line {
  background-color: red;
  border-radius: 3px;
}

div p {
  margin: 0;
  padding: 1em;
}

div p + p {
  padding-top: 0;
}
```

“注释掉”代码对于暂时禁用代码段进行测试也很有用。 `.special` 在下面的示例中，通过“注释掉”代码禁用了规则。

```
/*.special {
  color: red;
}*/

p {
  color: blue;
}
```

向您的 CSS 添加注释。

空白

空白意味着实际的空格、制表符和新行。就像浏览器会忽略 HTML 中的空格一样，浏览器也会忽略 CSS 中的空格。空白的价值在于它如何提高可读性。

在下面的示例中，每个声明（和规则开始/结束）都有自己的行。这可以说是一种编写 CSS 的好方法。它使维护和理解 CSS 变得更加容易。

```
body {
  font: 1em/150% Helvetica, Arial, sans-serif;
  padding: 1em;
  margin: 0 auto;
  max-width: 33em;
}

@media (min-width: 70em) {
  body {
    font-size: 130%;
  }
}

h1 {
  font-size: 1.5em;
}

div p,
#id:first-line {
  background-color: red;
  border-radius: 3px;
}

div p {
  margin: 0;
  padding: 1em;
}

div p + p {
  padding-top: 0;
}
```

下一个示例以更压缩的格式显示等效的 CSS。尽管这两个示例的工作原理相同，但下面的示例更难阅读。

```
margin: 0 auto; max-width: 33em;}
@media (min-width: 70em) { body { font-size: 130%;}}

h1 {font-size: 1.5em;}

div p, #id:first-line {background-color: red; border-radius:
3px;}
div p {margin: 0; padding: 1em;}
div p + p {padding-top: 0;}
```

对于您自己的项目，您将根据个人喜好格式化代码。对于团队项目，您可能会发现一个团队或项目有自己的风格指南。

> **警告：**虽然 CSS 声明中的值由空格分隔，但**属性名称永远不会有空格**。

例如，这些声明是有效的 CSS：

```
margin: 0 auto;
padding-left: 10px;
```

但是这些声明是无效的：

```
  margin: 0auto;
  padding- left: 10px;
```

你看到间距错误了吗？首先， `0auto` 不被识别为该属性的有效值 `margin` 。该条目 `0auto` 是两个独立的值： `0` 和 `auto` 。其次，浏览器不识别 `padding-` 为有效属性。正确的属性名称（ `padding-left` ）由错误的空格分隔。

您应该始终确保至少用一个空格将不同的值彼此分开。将属性名称和属性值作为单个完整的字符串放在一起。

要了解间距如何破坏 CSS，请尝试在测试 CSS 中使用间距。

概括

此时，您应该对 CSS 的结构有了更好的了解。了解浏览器如何使用 HTML 和 CSS 来显示网页也很有用。下一篇文章CSS 的工作原理解释了这个过程。

此页面最后修改于 2023 年 3 月 2 日由MDN 贡献者提供。