mdn web docs

> Esta página foi traduzida do inglês pela comunidade.
>
> Saiba mais e junte-se à comunidade MDN Web Docs.

Como CSS é estruturado

Agora que você tem uma ideia sobre o que é o CSS e seu uso básico, é hora de olhar um pouco mais o fundo das estruturas da linguagem em si. Nós já conhecemos muitos conceitos discutidos aqui, entretanto, você pode voltar para qualquer um em específico, se achar algum dos próximos conceitos um tanto confuso

Pré-requisitos:	Conceitos básicos de computação, softwares básicos instalados , conhecimentos básicos de operação com arquivos , básico de HTML (veja Introdução ao HTML), e uma ideia de Como funciona o CSS(en-US) .

Objetivo:	Aprender as estruturas da sintaxe básica do CSS em detalhes.

Aplicando CSS no seu HTML

A primeira coisa que você vai olhar é os três métodos de aplicação do CSS em um documento.

Folha de Estilos Externos

Em Começando com o CSS nós linkamos uma folha de estilos externos em nossa página. Isso é o método mais comumente utilizado para juntar CSS em um documento, podendo utilizar tal método em múltiplas páginas, permitindo que você estilize todas as páginas como as mesmas folhas de estilos. Na maioria dos casos, as diferentes páginas do site parecerão bem iguais entre si e por isso você pode usar as mesmas regras para o estilo padrão da página.

`Uma folha de estilos externa é quando você tem seu CSS escrito em um arquivo separado com uma extensão .css` , e você o refere dentro de um elemento `<link>` do HTML:

HTML

```
<!doctype html>
<html>
  <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>
```

O arquivo CSS deve parecer algo nesse estilo:

CSS

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

p {
  color: red;
}
```

O atributo `href` do elemento `<link>`, precisa fazer referência a um arquivo em nosso sistema de arquivos.

No exemplo abaixo, o arquivo CSS está na mesma pasta que o documento HTML, mas você pode colocá-lo em outro lugar e reajustar o caminho marcado para encontrá-lo, como a seguir:

HTML

```
<!-- 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" />
```

Folha de estilos internos

Uma folha de estilos internos é usada quando você não tem um arquivo CSS externo, mas, ao contrário, coloca seu CSS dentro do elemento `<style>` localizado no `<head>` documento HTML.

Deste modo, seu HTML parecerá assim:

HTML

```
<!doctype html>
<html>
  <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>
```

Isso pode ser útil em algumas situações (talvez você esteja trabalhando em um sistema de gerenciamento de conteúdo - CMS - onde não tem permissão para modificar diretamente os arquivos CSS), entretanto, isso não é tão eficiente quanto ao uso de folhas de estilo externo — em Em um site, o CSS precisaria ser repetido em todas as páginas e atualizado em vários locais sempre que fossem necessárias alterações.

Estilos embutidos

Estilos inline são declarações CSS que afetam apenas um elemento específico, inserido em um atributo `style`:

HTML

```
<!doctype html>
<html>
  <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>
```

Por favor, não utilize isso a menos que seja estritamente necessário! É péssimo para manutenção (você precisará atualizar a mesma informação diversas vezes em cada documento), além do que, mistura sua informação de estilização do CSS com sua informação de estrutura HTML, tornando seu código de difícil leitura e compreensão. Manter diferentes tipos de código separados torna o trabalho muito mais fácil para todos os que trabalham no código.

Existem alguns lugares onde o estilo embutido é mais comum, ou mesmo aconselhável. Você pode ter que recorrer ao uso deles se seu ambiente de trabalho for realmente restritivo (talvez o seu CMS permita apenas que você edite o corpo do HTML). Você também os verá sendo muito usados em e-mails em HTML de

modo a obter compatibilidade com o maior número possível de clientes de e-mail.

Brincando com o CSS neste artigo

Há muito CSS para brincar neste artigo. Para fazê-lo, recomendamos criar um novo diretório/pasta em seu computador e, dentro dele, criar uma cópia dos seguintes dois arquivos:

index.html:

HTML

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

styles.css:

CSS

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

```
p {
  color: red;
}
```

Então, quando você encontrar algum CSS com o qual queira experimentar, substitua o conteúdo do <*body*> HTML por algum HTML para estilizar e comece a adicionar CSS para estilizá-lo dentro do seu arquivo CSS.

Se você não estiver usando um sistema em que possa criar arquivos facilmente, você pode usar o editor interativo abaixo para experimentar.

Continue lendo e divirta-se!

Seletores

Não é possível falar de CSS sem conhecer os seletores, e nós já descobrimos vários tipos diferentes no tutorial Começando com o Css. Um seletor é o modo pelo qual nós apontamos para alguma coisa no nosso documento HTML para aplicar os estilos à ela. Se os seus estilos não forem aplicados, então é provável que o seu seletor não esteja ligado aquilo que você pensa que ele deveria.

Cada regra CSS começa com um seletor ou uma lista de seletores para informar ao navegador em qual elemento ou elementos as regras devem ser aplicadas. Todos os exemplos a seguir são válidos como seletores ou listas de seletores.

CSS

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

Tente criar algumas regras CSS que usem os seletores acima e algum HTML para ser estilizado por eles. Se você não souber o que significa alguma das sintaxes acima, tente procurar no MDN!

> **Nota:** Você aprenderá muito mais sobre seletores em nossos tutoriais CSS selectors, no próximo módulo.

Especificidade

Muitas vezes, haverá cenários em que dois seletores podem selecionar o mesmo elemento HTML. Considere a folha de estilo abaixo, onde há uma regra com um seletor p que definirá parágrafos como azuis e também uma classe que definirá elementos selecionados como vermelhos.

CSS

```
.special {
  color: red;
}

p {
  color: blue;
}
```

Digamos que em nosso documento HTML tenhamos um parágrafo com uma classe `special` . Ambas as regras poderiam

ser aplicadas, então qual delas vence? Qual cor você acha que nosso parágrafo ficará?

HTML

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

A linguagem CSS possui regras para controlar qual regra vencerá em caso de colisão - elas são chamadas de **cascata** e **especificidade**. No bloco de código abaixo, definimos duas regras para o seletor p, mas o parágrafo acaba sendo colorido de azul. Isso ocorre porque a declaração que o define como azul aparece posteriormente na folha de estilo, e estilos mais recentes substituem os anteriores. Isso é a cascata em ação.

CSS

```
p {
  color: red;
}

p {
  color: blue;
}
```

No entanto, no caso do nosso bloco anterior com o seletor de classe e o seletor de elemento, a classe vencerá, tornando o parágrafo vermelho - mesmo que apareça anteriormente na folha de estilo. Uma classe é descrita como sendo mais específica, ou

tendo mais especificidade do que o seletor de elemento, então ela vence.

Experimente o exemplo acima por si mesmo - adicione o HTML ao seu experimento, em seguida, adicione as duas regras p {...} à sua folha de estilo. Em seguida, altere o primeiro seletor p para .special para ver como ele muda o estilo.

As regras de especificidade e cascata podem parecer um pouco complicadas no início e são mais fáceis de entender depois que você tiver acumulado mais conhecimento de CSS. Em nosso artigo Cascade and inheritance, que você verá no próximo módulo, explicarei isso em detalhes, incluindo como calcular a especificidade. Por enquanto, lembre-se de que isso existe e que às vezes o CSS pode não ser aplicado como você espera porque algo mais em sua folha de estilo tem uma especificidade maior. Identificar que mais de uma regra pode ser aplicada a um elemento é o primeiro passo para resolver esses problemas.

Propriedades e valores

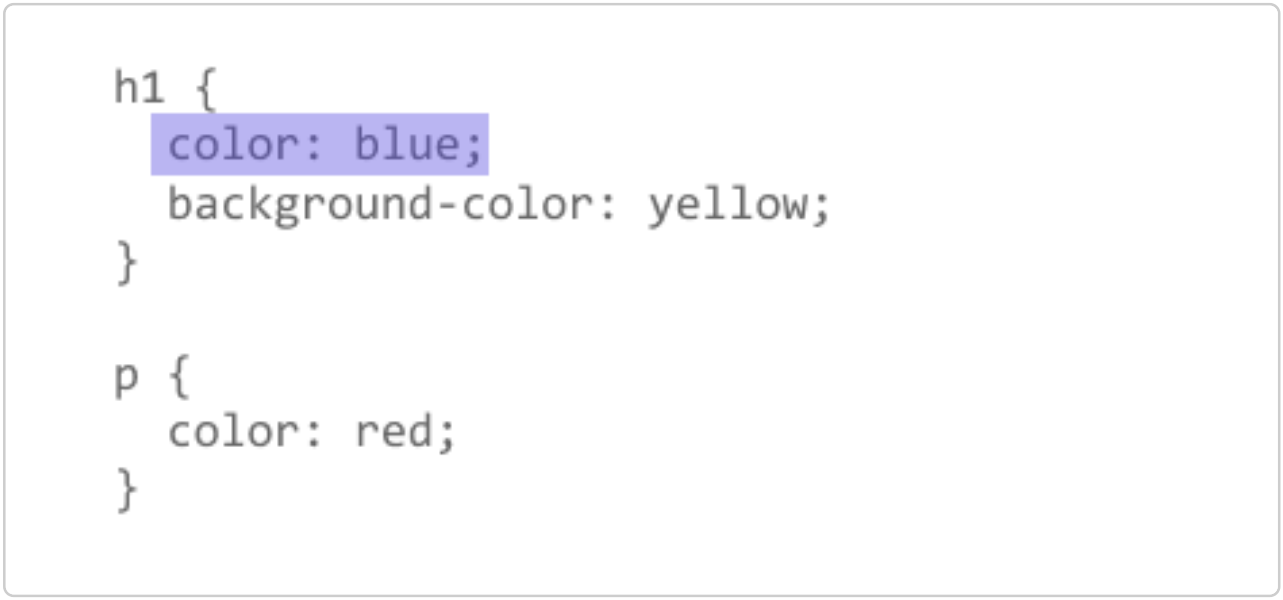
Em seu nível mais básico, CSS consiste em dois blocos de construção:

- **Properties**: Identificadores legíveis para humanos que indicam quais características estilísticas (por exemplo, `font-`

`size` , `width` , `background-color`) que você deseja alterar.

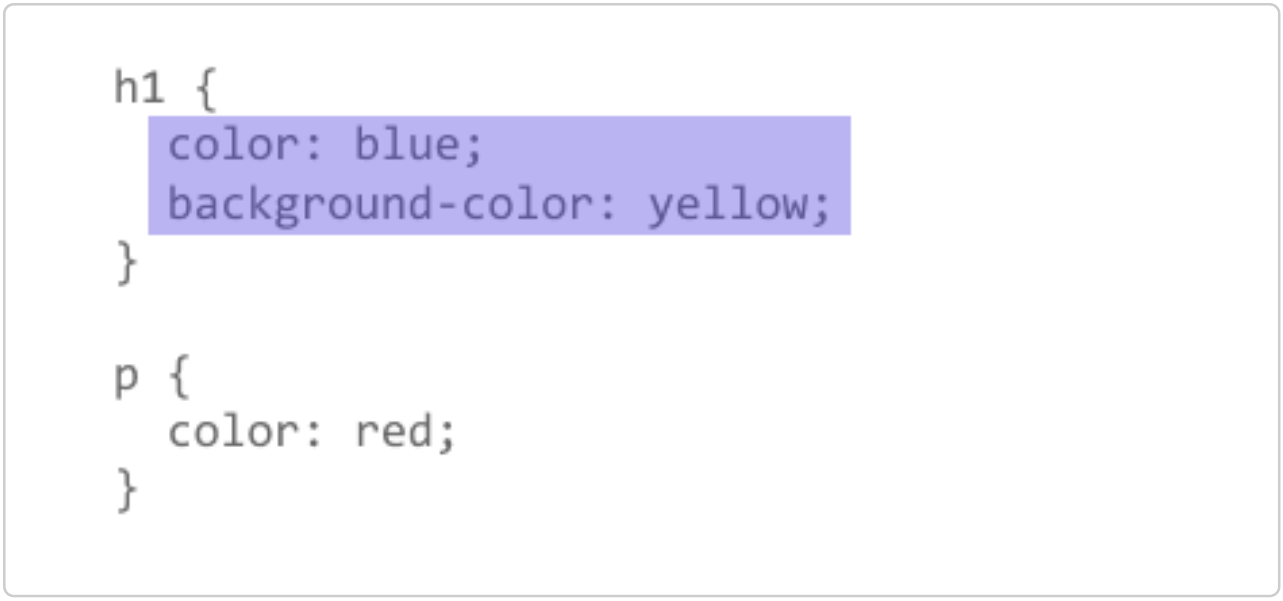
- **Valores**: Cada propriedade especificada recebe um valor, que indica como você deseja alterar essas características estilísticas (por exemplo, o que deseja mudar a fonte, a largura ou a cor de fundo para).

A imagem abaixo destaca uma única propriedade e valor. O nome da propriedade é `color` e o valor é `blue` .

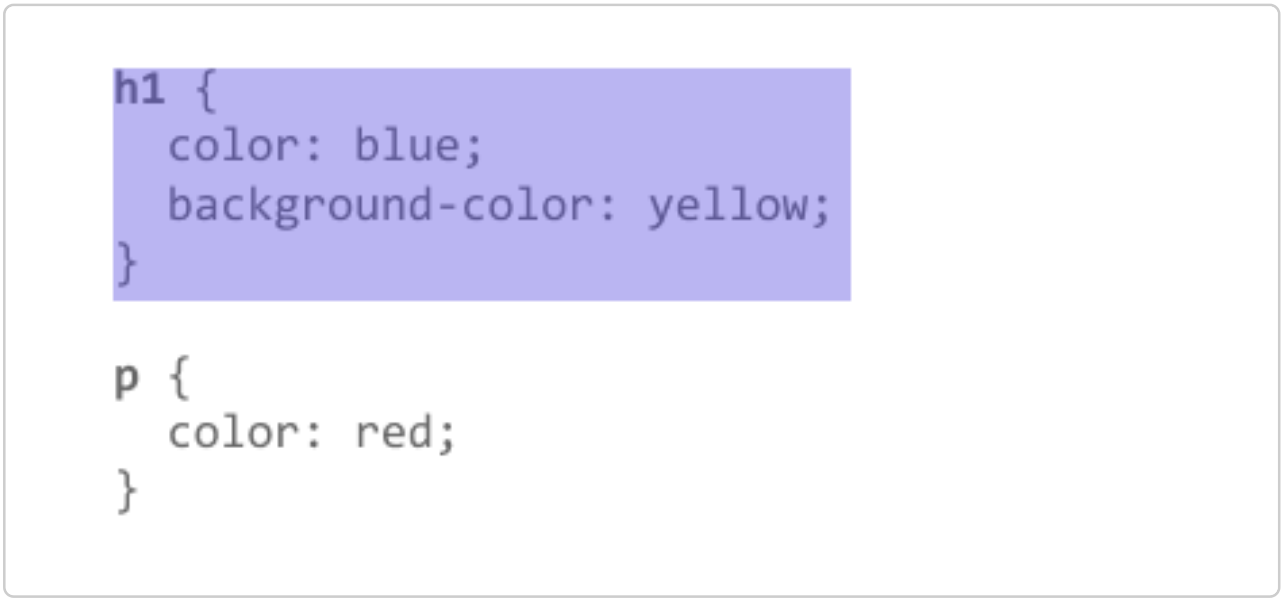


A property paired with a value is called a *CSS declaration*. CSS declarations are put within *CSS Declaration Blocks*. This next image shows our CSS with the declaration block highlighted.

Uma propriedade associada a um valor é chamada de *Declaração CSS*. As declarações CSS são colocadas dentro de *Blocos de Declaração CSS*. A próxima imagem mostra nosso CSS com o bloco de declaração destacado.



Por fim, blocos de declarações CSS são pareados com *seletores* para produzir *Conjuntos de Regras CSS* (ou *Regras CSS*). Nossa imagem contém duas regras, uma para o seletor `h1` e outra para o seletor `p`. A regra para o seletor h1 está destacada.



Definir propriedades CSS para valores específicos é a função central da linguagem CSS. O motor CSS calcula quais declarações se aplicam a cada elemento de uma página para ajustá-la e estilizá-la adequadamente. O que é importante

lembrar é que tanto as propriedades quanto os valores diferenciam letras maiúsculas e minúsculas em CSS. A propriedade e o valor em cada par são separados por dois pontos (:).

Tente procurar diferentes valores das seguintes propriedades e escrever regras CSS que as apliquem a diferentes elementos HTML:

- `font-size`
- `width`
- `background-color`
- `color`
- `border` (en-US)

> **Aviso: Importante**: Se uma propriedade é desconhecida ou se um valor não é válido para uma determinada propriedade, a declaração é considerada *inválida* e é completamente ignorada pelo motor CSS do navegador.

> **Aviso: Importante**: Em CSS (e em outros padrões da web), a ortografia americana foi estabelecida como padrão a ser seguido quando houver incerteza

> linguística. Por exemplo, `color` deve ser *sempre* escrito como `color`. `colour` não funcionará.

Funções

Embora a maioria dos valores sejam palavras-chave relativamente simples ou valores numéricos, existem alguns valores possíveis que assumem a forma de uma função. Um exemplo seria a função `calc()`. Essa função permite que você faça cálculos simples dentro do seu CSS, por exemplo:

HTML Play

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

CSS Play

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

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

Isto é renderizado assim:

Play

Uma função consiste no nome da função e, em seguida, em alguns parênteses nos quais os valores permitidos para essa função são inseridos. No exemplo `calc()` acima, estou pedindo para a largura desta caixa ser 90% da largura do bloco contenedor, menos 30 pixels. Isso não é algo que eu possa calcular com antecedência e simplesmente inserir o valor no CSS, pois não sei qual será 90%. Como em todos os valores, a página relevante no MDN terá exemplos de uso para que você possa ver como a função funciona.

Outro exemplo seriam os vários valores para `transform`, como `rotate()`.

HTML Play

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

CSS Play

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

A saída do código acima se parece com isso:

Play

Tente procurar diferentes valores das seguintes propriedades, e escreva regras CSS que as apliquem a diferentes elementos HTML:

- `transform`
- `background-image` (en-US), in particular gradient values

- `color` , **in particular rgb/rgba/hsl/hsla values**

@rules

Até agora, não encontramos as regras, em inglês `@rules` (pronuncia-se "at-rules") do CSS. Estas são regras especiais que dão ao CSS algumas instruções sobre como se comportar. Algumas `@rules` são simples, com o nome da regra e um valor. Por exemplo, para importar uma folha de estilo adicional na sua folha de estilo CSS principal, você pode usar `@import` :

CSS

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

Uma das `@rules` mais comuns que você encontrará é a `@media` , que permite usar media queries para aplicar CSS somente quando certas condições são verdadeiras (por exemplo, quando a resolução da tela está acima de um certo valor ou a tela é mais larga que uma largura específica).

Nesse CSS abaixo, temos um estilo que dá ao elemento `<body>` uma cor de fundo rosa. No entanto, usamos `@media` para criar uma seção do nosso estilo que será aplicada apenas em navegadores com uma viewport mais larga que 30em. Se o navegador for mais largo que 30em, a cor de fundo será azul.

CSS

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

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

Você encontrará outras `@rules` durante estes tutoriais.

Veja se você consegue adicionar uma consulta de mídia ao seu CSS que altere estilos com base na largura da viewport. Altere a largura da janela do seu navegador para ver o resultado.

Atalhos

Algumas propriedades como `font` (en-US), `background`, `padding`, `border` (en-US) e `margin` são chamadas de **propriedades abreviadas** - isso porque elas permitem que você defina vários valores de propriedade em uma única linha, economizando tempo e tornando seu código mais organizado no processo.

Por exemplo, está linha:

```
CSS
```

```
/* 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;
```

Faz a mesma coisa que todos eles juntos:

CSS

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

Por exemplo, está linha:

CSS

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

Faz a mesma coisa que todos eles juntos:

CSS

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

```
background-repeat: repeat-x;
background-scroll: fixed;
```

Não tentaremos ensinar isso exaustivamente agora - você encontrará muitos exemplos mais tarde no curso, e é aconselhável procurar os nomes das propriedades abreviadas em nossa Referência de CSS para saber mais.

Tente adicionar as declarações acima ao seu CSS para ver como elas afetam o estilo do seu HTML. Tente experimentar com alguns valores diferentes.

> **Aviso:** Embora os atalhos geralmente permitam que você deixe de fora valores, eles então redefinem quaisquer valores que você não incluir para seus valores iniciais. Isso garante que um conjunto sensato de valores seja usado. No entanto, isso pode ser confuso se você estiver esperando que o atalho apenas mude os valores que passou.

Comentários

Assim como no HTML, você é incentivado a fazer comentários em seu CSS, para ajudá-lo a entender como seu código funciona quando voltar a ele depois de vários meses, e para ajudar outras

pessoas a entenderem o código quando estiverem trabalhando nele.

Comentários em CSS começam com `/*` e terminam com `*/`. No bloco de código abaixo, foram usados comentários para marcar o início de diferentes seções de código distintas. Isso é útil para ajudar na navegação da sua base de código à medida que ela cresce - você pode procurar pelos comentários no seu editor de código.

CSS

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

@media (min-width: 70em) {
  /* Let's special case the global font size. On large screen or
window,
     we increase the font size 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;
  background-style: none;
}

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

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

Os comentários também são úteis para *comentar* temporariamente partes do código para fins de teste, por exemplo, se você estiver tentando encontrar qual parte do seu código está causando um erro. No próximo exemplo, comentei as regras para o seletor `.special`.

CSS

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

p {
  color: blue;
}
```

Adicione alguns comentários ao seu CSS, para se acostumar a usá-los.

Espaçamento

Espaços em branco significam espaços, tabulações e novas linhas. Da mesma forma que no HTML, o navegador tende a ignorar grande parte do espaçamento dentro do seu CSS; grande parte do espaçamento está presente apenas para ajudar na legibilidade.

No exemplo abaixo, temos cada declaração (e início/fim de regra) em sua própria linha - essa é uma maneira recomendada de escrever CSS, já que torna fácil manter e entender:

CSS

```
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;
  background-style: none;
}

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

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

Você poderia escrever exatamente o mesmo CSS como abaixo, com grande parte do espaçamento removido - isto é, funcionalmente idêntico ao primeiro exemplo, mas é mais difícil de ler:

CSS

```css
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;
  background-style: none;
}
div p {
```

```
  margin: 0;
  padding: 1em;
}
div p + p {
  padding-top: 0;
}
```

O layout de código que você escolher geralmente é uma preferência pessoal, embora quando você começar a trabalhar em equipes, pode descobrir que a equipe existente tem seu próprio guia de estilo que especifica uma convenção acordada a seguir.

O espaçamento que você precisa ter cuidado no CSS é o espaçamento entre as propriedades e seus valores. Por exemplo, as seguintes declarações são CSS válidos:

CSS

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

Mas os seguintes são inválidos:

CSS

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

`0auto` não é reconhecido como um valor válido para a propriedade `margin` (`0` e `auto` são dois valores separados), e o navegador não reconhece `padding-` como uma propriedade válida. Portanto, você sempre deve garantir que valores distintos estejam separados um do outro por pelo menos um espaço, mas mantenha os nomes das propriedades e os valores das propriedades juntos como uma única cadeia ininterrupta.

Experimente brincar com espaçamento dentro do seu CSS, para ver o que quebra as coisas e o que não quebra.

Qual é o próximo passo?

É útil entender um pouco sobre como o navegador pega seu HTML e CSS e transforma-o em uma página da web, então no próximo artigo — Como CSS Funciona — vamos dar uma olhada nesse processo.

This page was last modified on 3 de ago. de 2023 by MDN contributors.