

Unidade III

5 ESTAÇÃO CSS3

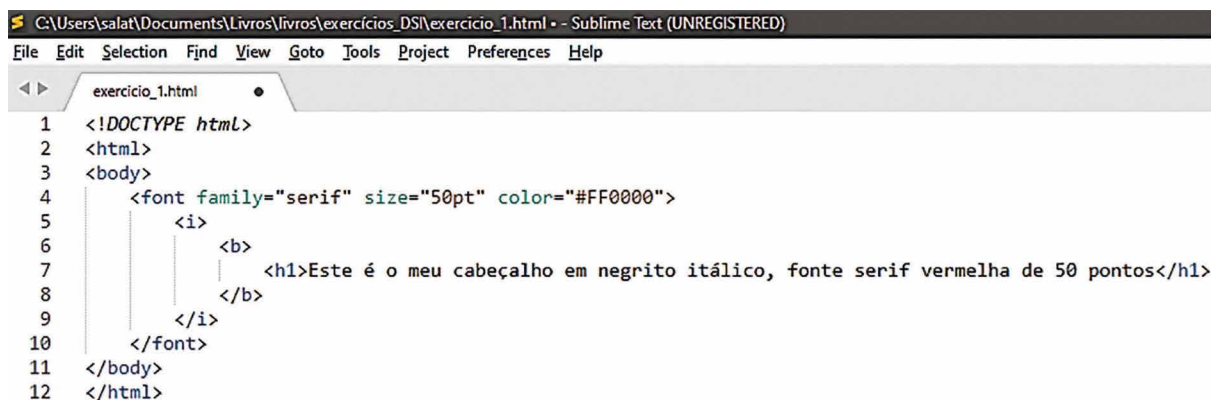
5.1 Introdução ao CSS3

CSS vem de *cascading style sheets*, uma linguagem capaz de estilizar (ou seja, efetuar a gestão de cores, tamanho, layout, posição etc.) o conteúdo de um documento de marcação HTML. Assim, embora o objetivo do código HTML seja estruturar o documento, o CSS tem como meta definir como esse conteúdo é apresentado ao usuário final da web.

Podemos incorporar o CSS ao próprio documento HTML, como já havíamos feito em alguns tópicos das unidades anteriores, em que criamos um documento de extensão .html, mas utilizando alguns comandos CSS incorporados internamente. Para isso, usamos o elemento <style> do cabeçalho do nosso documento ou o atributo style de elementos específicos.

No entanto, um recurso muito interessante dessa linguagem de estilos é a possibilidade de criar os próprios documentos CSS. Esses documentos, com extensão .css, são arquivos separados do documento HTML, que contêm as instruções de estilo que devem ser adotadas pelas nossas páginas web. Dessa forma, referenciamos nossos arquivos .html a arquivos externos .css.

O CSS foi originalmente gerado para garantir a redução da quantidade de marcação que estava sendo utilizada em um documento HTML, ditando, assim, a aparência de um trecho específico de HTML, conforme apresentado a seguir.



```
1 <!DOCTYPE html>
2 <html>
3 <body>
4 <font family="serif" size="50pt" color="#FF0000">
5 <i>
6 <b>
7 <h1>Este é o meu cabeçalho em negrito itálico, fonte serif vermelha de 50 pontos</h1>
8 </b>
9 </i>
10 </font>
11 </body>
12 </html>
```

Figura 91 – Exemplo de código pré-CSS

Como podemos ver no código HTML apresentado na figura anterior, é necessário muito mais marcação para produzir um cabeçalho simples, em uma linha, em negrito e vermelho, do que na maioria dos códigos que são utilizados atualmente.

***Este é o meu cabeçalho em
negrito itálico, fonte serif
vermelha de 50 pontos***

Figura 92 – A saída produzida pelo exemplo de código pré-CSS

As regras CSS resolvem dois desses problemas imediatamente apresentados no código. Em primeiro lugar, geralmente apresentam-se as regras que compõem a aparência do texto em um arquivo separado. Isso significa que um designer ou qualquer pessoa da área gráfica pode reajustar facilmente essas regras, sem ter necessidade de se preocupar com o código do arquivo HTML. Em segundo lugar, consegue-se promover a reutilização de uma regra em diversos elementos, o que significa que podemos especificar as configurações de estilo somente uma vez e informar a marcação quando utilizá-las.

Assim como as novas especificações do HTML5, o CSS3 (ou Cascading Style Sheets versão 3, para ser mais preciso) é definido como um conjunto mais recente de especificações projetadas para efetuar o estilo de páginas web. O CSS3 também não é um novo CSS, como muitas pessoas pensam. Assim como o HTML5, ele é retrocompatível com tudo o que veio antes. No futuro, no entanto, haverá uma incrível quantidade de novas funcionalidades, permitindo uma infinidade de possibilidades criativas que não existiam nas versões anteriores.

Ao longo desta unidade, vamos estudar como associar o CSS aos nossos documentos HTML. Apenas para uma demonstração inicial, no quadro a seguir, vemos um exemplo de regra de estilo em CSS, que traz as características do título da figura anterior em vermelho e negrito. Podemos criar um arquivo, em qualquer editor de texto, que deve ser salvo com a extensão .css. Esse arquivo CSS contém uma classe chamada **boldred**, que especifica a regra de estilo que queremos adotar.

Quadro 10 – Classe boldred

.boldred
{
font-family: serif;
font-size: 20pt;
color: red;
}

Posteriormente, podemos adotar essa regra de estilo dentro do código HTML, chamando-a por meio do atributo `class`. Se decidirmos, em algum momento, alterar a cor da fonte especificada na regra de vermelho para azul, todos os elementos que utilizarem essa regra sofrerão alterações para a cor azul, sem precisar alterar o código HTML. Por conta disso, o CSS é uma poderosa ferramenta. Vamos começar a parte prática do nosso estudo.

5.2 Sintaxe básica do CSS

Uma regra de estilo CSS é formada, basicamente, por um **seletor** e por um **bloco de declaração**. O seletor aponta qual elemento HTML deverá ser estilizado. Já o bloco de declaração contém uma ou mais declarações, que indicam propriedades de estilo, separadas entre si por ponto e vírgula. Vamos acompanhar um exemplo simples.

```
h2 {  
  color: blue;  
  text-align: right;  
}
```

Nesse trecho de código CSS, vemos o seletor `h2`, indicando que o elemento HTML a ser estilizado é o `<h2>`. O bloco de declaração está delimitado pelas chaves. Dentro das chaves, vemos a propriedade `color` (que assumiu o valor *blue*) e a propriedade `text-align` (que assumiu o valor *right*). Dessa forma, todo o texto do documento HTML que aparece entre as marcas `<h2>` e `</h2>` deve ser mostrado em cor azul e com alinhamento à direita.

Uma folha de estilos em cascata pode conter diversas regras de estilo, não apenas uma, como visto no exemplo anterior. Poderíamos, por exemplo, ter criado uma regra para o elemento `<h2>`, outra regra para o `<h3>` e uma terceira regra para o `<p>`. Qualquer elemento pode ser estilizado dessa forma. Podemos entender uma folha de estilos, portanto, como um conjunto de regras de estilo que formatarão a aparência de uma página HTML.

5.3 Formas de adição de CSS

Vamos agora aplicar o estilo básico visto no tópico anterior a uma página web de três formas:

- criar um arquivo CSS separado (CSS externo);
- utilizar a incorporação do estilo no arquivo HTML (CSS interno);
- usar o recurso de CSS em linha (CSS inline).

CSS externo

Podemos criar um arquivo `.css` externo, que deverá se comunicar com nosso arquivo `.html`, de acordo com o demonstrado nas figuras seguintes. Esse formato de adição de CSS é conhecido como CSS externo e é considerado um método muito eficiente de estilizar websites grandes.



Figura 93 – Arquivo CSS style1.css



Figura 94 – Arquivo HTML que se comunica com o arquivo style1.css

Para criar esses arquivos, vamos seguir os passos descritos:

- Abra o Sublime Text.
- Com o editor aberto, clique em "File" > "Save As...".
- A janela Salvar Como deve estar aberta na sua tela. Selecione a pasta de destino, digite o nome do arquivo como style1.css e escolha o tipo como CSS, conforme o que segue.

Nome: style1.css

Tipo: CSS (*.css;*.css.erb;*.css.liquid)

- Clique em "Salvar".
- Você deve voltar para a tela de edição de texto do Sublime Text, já enxergando o nome do arquivo na aba superior. Agora, digite o nosso código CSS. Ele será a nossa folha de estilos que, no caso, contém apenas uma regra de estilo.

```
h2 {  
  color: blue;  
  text-align: right;  
}
```

- Salve o arquivo. Com isso, concluímos a nossa folha de estilos, salva no arquivo CSS.
- Vamos, então, para o código HTML. Crie um arquivo HTML da forma convencional, mas com a inserção de uma referência para o arquivo CSS em seu cabeçalho (entre as marcas <head> e </head>). Essa referência é feita por meio da tag <link>, contendo a seguinte sintaxe:

```
<link rel="stylesheet" type="text/css" href="style1.css">
```

- O valor do atributo **href** deve conter o nome da sua folha de estilos, seja ele qual for. Isso é válido para os casos em que o arquivo CSS se encontra no mesmo diretório do arquivo HTML. Caso esteja em um local diferente, o caminho da folha de estilos deve ser especificado.
- Podemos também utilizar a sintaxe do XHTML para a tag <link> (é bastante comum encontrarmos documentações escritas dessa forma). Nesse caso, temos:

```
<link rel="stylesheet" type="text/css" href="style1.css" />
```

- Conclua o código HTML e salve o arquivo.
- A execução no navegador deve trazer um resultado similar ao visto na figura a seguir.



Figura 95 – Saída do exemplo de sintaxe básica CSS utilizando folha de estilo externa

Existe a opção de verificar a execução da página acionando o botão F12 do teclado e visualizando as interações da página, conforme apresentado a seguir.

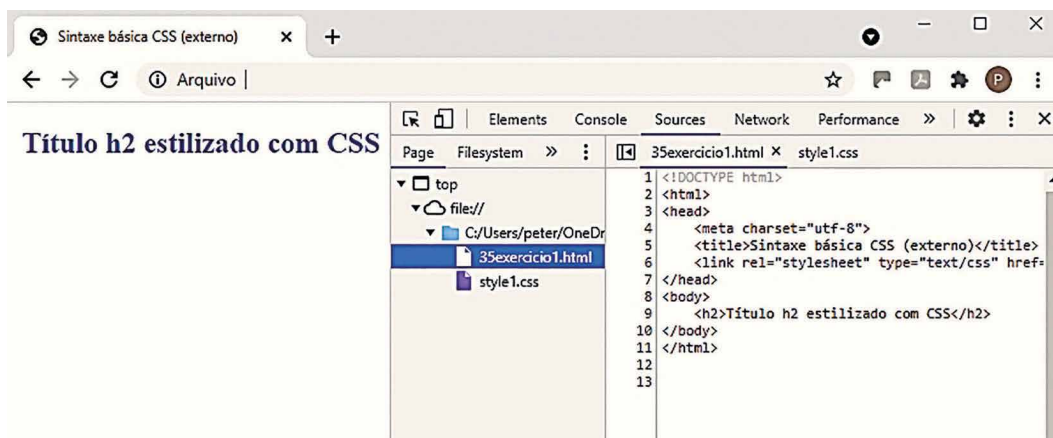


Figura 96 – Ferramentas do navegador Google Chrome mostrando o arquivo CSS vinculado ao arquivo HTML

No caso do CSS externo, se a página HTML não localizar um arquivo CSS, nenhuma das regras que se encontram no arquivo de estilo será aplicada a ela. Experimente, por exemplo, modificar o valor do atributo **href** para **href="style2.css"**. Como esse arquivo não existe no diretório, a regra de estilo especificada por nós para o elemento **<h2>** será substituída pela formatação padrão do navegador.

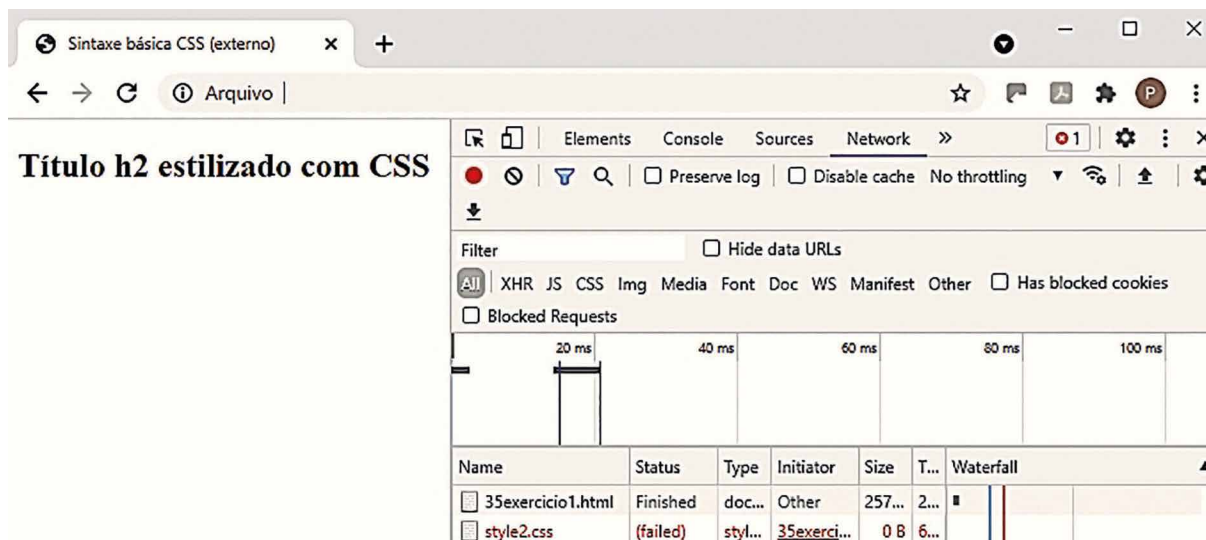


Figura 97 – Ferramentas do navegador Google Chrome mostrando o arquivo CSS falhando ao carregar



Saiba mais

A utilização do browser Google Chrome é uma recomendação. Para realizar a instalação desse browser, acesse o link na indicação a seguir.

Disponível em: <https://bit.ly/3N0mEnO>. Acesso em: 10 maio 2022.

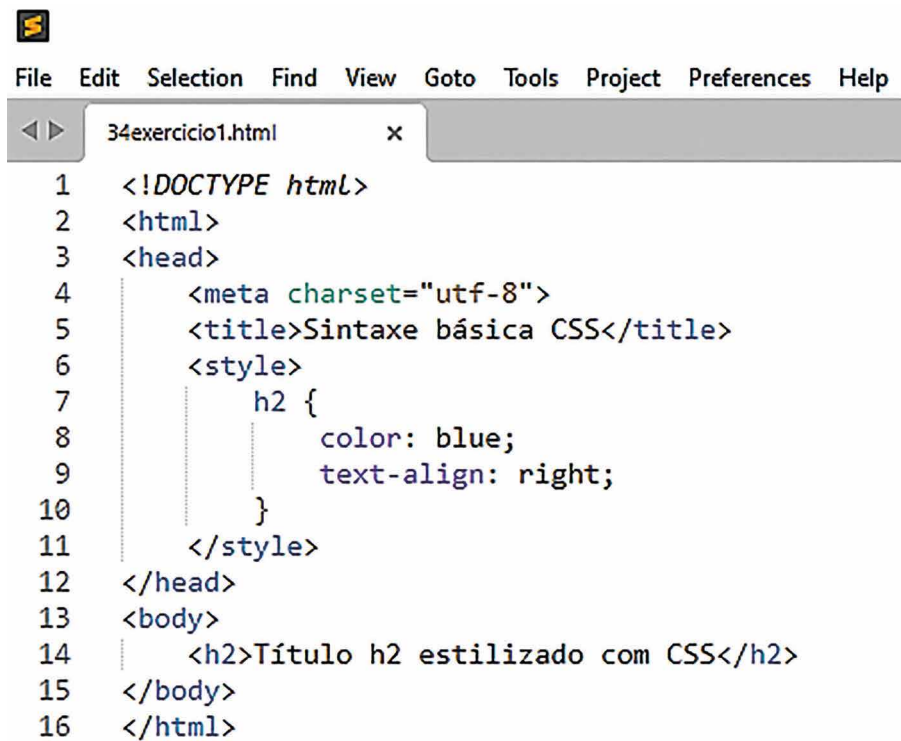
O artigo a seguir nos dá uma explanação sobre a utilização do modo desenvolvedor do browser Google Chrome.

USE as ferramentas para desenvolvedores do Google Chrome para verificar tags. Ajuda do Campaign Manager 360. *Support Google*, 2020. Disponível em: <https://bit.ly/3N5yehG>. Acesso em: 10 maio 2022.

CSS interno

Se incorporarmos a nossa folha de estilos CSS ao cabeçalho do arquivo HTML, ela deve aparecer entre as tags **<style>** e **</style>**, conforme demonstrado na figura a seguir. Esse formato de adição de CSS é conhecido como CSS interno e é adequado quando a intenção é estilizar uma única página HTML.

Nesse caso, não há criação de um arquivo separado, pois toda a folha de estilos CSS está contida no próprio arquivo HTML.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Sintaxe básica CSS</title>
6     <style>
7         h2 {
8             color: blue;
9             text-align: right;
10        }
11    </style>
12 </head>
13 <body>
14     <h2>Título h2 estilizado com CSS</h2>
15 </body>
16 </html>
```

Figura 98 – Exemplo de sintaxe básica CSS utilizando folha de estilo interna



Figura 99 – Saída do exemplo de sintaxe básica CSS utilizando folha de estilo interna

CSS inline

Podemos, ainda, utilizar outro formato de incorporação de CSS a um código HTML, conhecido como CSS *inline* (em linha). Quando utilizamos esse formato, especificamos estilos individualmente aos elementos HTML por meio do atributo `style` do elemento. Nesse caso, o código também é incorporado ao próprio arquivo HTML, sem a criação de um arquivo separado.

Esse método é o menos versátil, pois, por meio dele, é necessário estilizar cada tag individualmente. De qualquer maneira, ele pode ser utilizado, por exemplo, quando um estilo é aplicado especificamente a um elemento, e não deve ser aproveitado por nenhum outro elemento da página.

Acompanhe o exemplo da figura a seguir. Note que não há a criação de uma folha de estilos no cabeçalho, como fizemos no método CSS interno, e sim a adoção de uma regra individual, por meio do atributo `style` do próprio elemento `h2`.



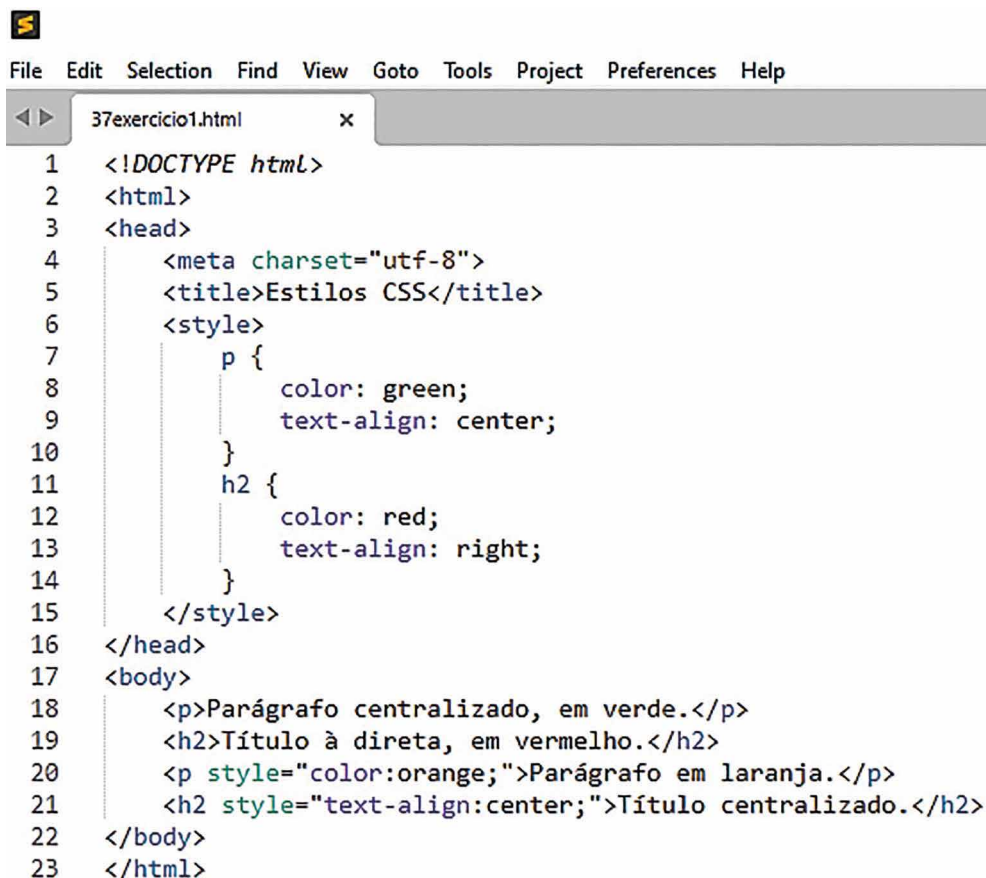
```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Sintaxe básica CSS (inline)</title>
6 </head>
7 <body>
8   <h2 style="color:blue;text-align:right;">Título h2 estilizado com CSS</h2>
9 </body>
10 </html>
```

Figura 100 – Exemplo de sintaxe básica CSS com regra de estilo inline



Figura 101 – Saída do exemplo de sintaxe básica CSS utilizando regra de estilo inline

O CSS *inline* tem prioridade em relação aos métodos interno e externo. Por exemplo, caso haja uma folha de estilos no cabeçalho para o elemento `<p>`, mas o atributo de um elemento possui outra regra *inline*, a regra em linha será utilizada. Observe o exemplo a seguir.



```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Estilos CSS</title>
6      <style>
7          p {
8              color: green;
9              text-align: center;
10         }
11         h2 {
12             color: red;
13             text-align: right;
14         }
15     </style>
16 </head>
17 <body>
18     <p>Parágrafo centralizado, em verde.</p>
19     <h2>Título à direita, em vermelho.</h2>
20     <p style="color:orange;">Parágrafo em laranja.</p>
21     <h2 style="text-align:center;">Título centralizado.</h2>
22 </body>
23 </html>
    
```

Figura 102 – Exemplo de código CSS com regras de estilo interna e inline

Há uma folha de estilos entre as marcas `<style>` e `</style>` no cabeçalho, indicando cor verde e alinhamento centralizado para o seletor `p`, assim como cor vermelha e alinhamento à direita para o seletor `h2`. Essas regras foram adicionadas por CSS interno. Já no corpo do documento (entre as marcas `<body>` e `</body>`) existem quatro elementos. Os dois primeiros, que aparecem sem nenhuma regra inline, buscaram informações de estilo direto do cabeçalho. Já os dois últimos adotaram, primeiramente, a regra *inline*, para depois adotarem as propriedades do cabeçalho. O resultado no navegador é mostrado a seguir.

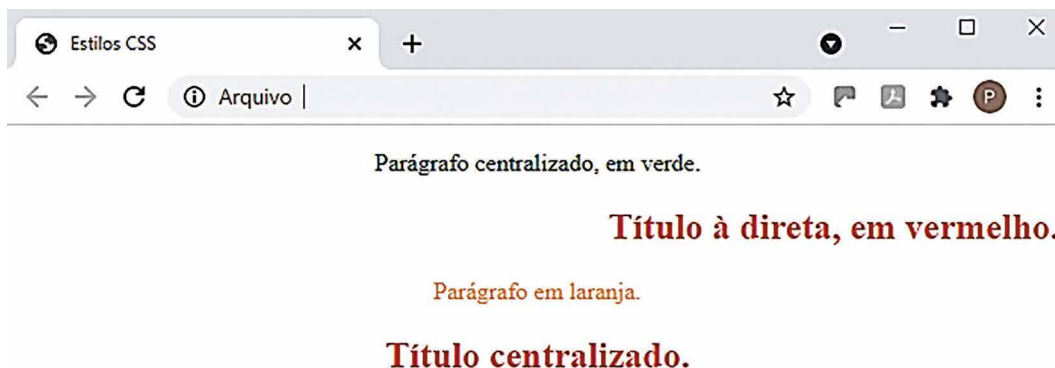


Figura 103 – Saída do exemplo com regras de estilo interna e inline

Observe que o segundo parágrafo aparece em laranja (de acordo com a regra inline) e com alinhamento centralizado (de acordo com a regra interna). Já o segundo título aparece em vermelho (de acordo com a regra interna) e com alinhamento centralizado (de acordo com a regra inline). Todas as propriedades não especificadas por nós (como tipo de fonte, tamanho de fonte etc.) são adotadas de acordo com o padrão do navegador.

5.4 Seletores CSS

Já aprendemos que um seletor CSS aponta qual elemento HTML deverá ser estilizado. No entanto, existem diversos tipos de seletores, que serão estudados ao longo deste tópico. Veremos seletores dos seguintes tipos:

- simples;
- combinados;
- de atributos;
- de pseudoclasses.

Por questões didáticas, adotaremos a técnica de CSS interno nos exemplos seguintes do nosso livro-texto. Dessa forma, é mais fácil avaliar tanto a folha de estilos quanto o código HTML.

5.4.1 Seletores simples

Primeiramente, estudaremos tipos básicos de seletores, que chamaremos de seletores simples. São eles:

- seletores diretos;
- seletores de ID;
- seletores de classe;
- seletor universal;
- agrupamento de seletores.

O entendimento dos seletores simples nos auxiliará a compreender tipos mais complexos posteriormente.

5.4.1.1 Seletores diretos

O seletor direto é o mais simples de todos os seletores disponibilizados. É utilizado para selecionar um elemento HTML conhecido diretamente pelo seu nome. Foi esse o seletor que usamos nos exemplos dos tópicos anteriores.

Seletores diretos também são conhecidos como seletores de elemento ou seletores de tag.

Podemos utilizar um seletor direto quando nos referimos diretamente a uma entidade nomeada, como p, h1, h2 etc. Observe a figura a seguir.

```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <style>
5 p
6 {
7   color: green;
8 }
9 </style>
10 </head>
11 <body>
12 <p>Unide 3</p>
13 </body>
14 </html>
```

Figura 104 – Exemplo para apresentar o código CSS com seletor direto

Depois que esse item é adicionado a uma folha de estilos (CSS), qualquer item na marcação HTML do documento que existir entre as marcas <p> e </p> será apresentado em cor verde, conforme ilustrado nas linhas 4 a 9, independentemente de sua classe, de seu ID ou de qualquer outra situação.



Lembrete

Aplica-se um estilo de cor verde para o elemento <p>, ou seja, todo texto inserido na página envolvido com ela será apresentado nessa cor.

5.4.1.2 Seletores de ID

Os seletores de ID utilizam o atributo **id** de elementos HTML para selecionar elementos específicos. O ID de um elemento deve ser único em uma página HTML, pois ele representa a identificação da tag. Dessa forma, uma regra criada para um elemento não deve ser reutilizada por outro. Para criarmos esse tipo de seletor, utilizamos o símbolo # seguido do ID do elemento.

É possível utilizar seletores de IDs para estilizar partes únicas da estrutura da sua página, como cabeçalho, rodapé, menu etc.

Acompanhe o exemplo a seguir, em que utilizamos seletores de ID.

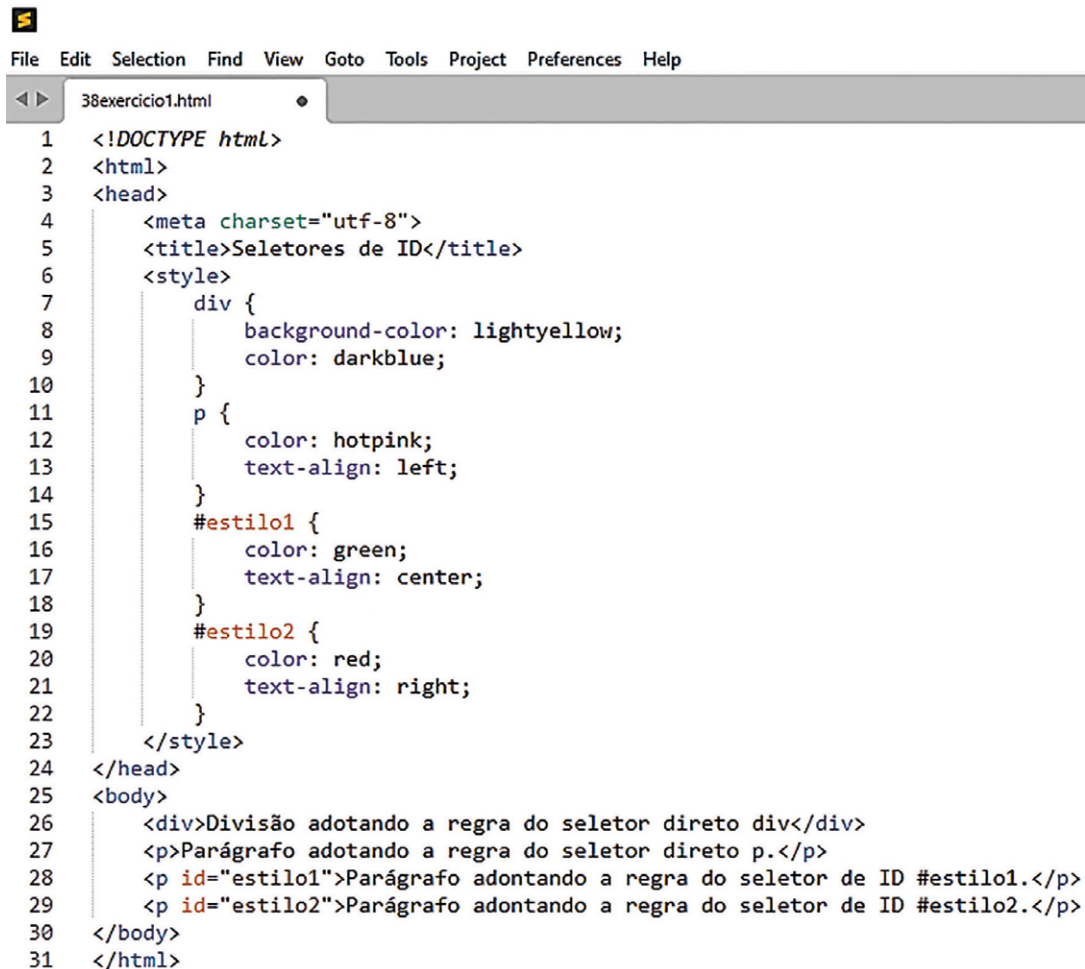


Figura 105 – Exemplo de código com seletores de ID

Nesse caso, os elementos <div> e <p> adotam as regras de seus seletores diretos, identificados na folha de estilos do cabeçalho. Note que ajustamos uma propriedade de estilo diferente das utilizadas até então: mudamos a cor do fundo do conteúdo do elemento <div> para amarelo-claro. Já os parágrafos que possuem uma identificação por meio de seu atributo id priorizam as regras designadas para seus identificadores. O resultado no navegador pode ser visto na figura seguinte.

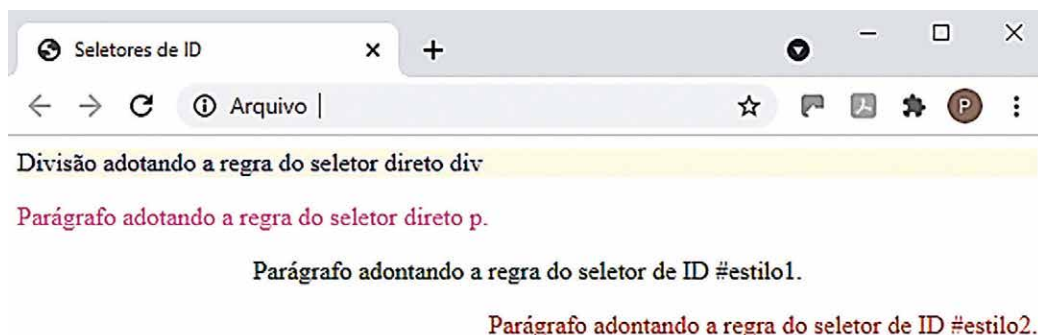
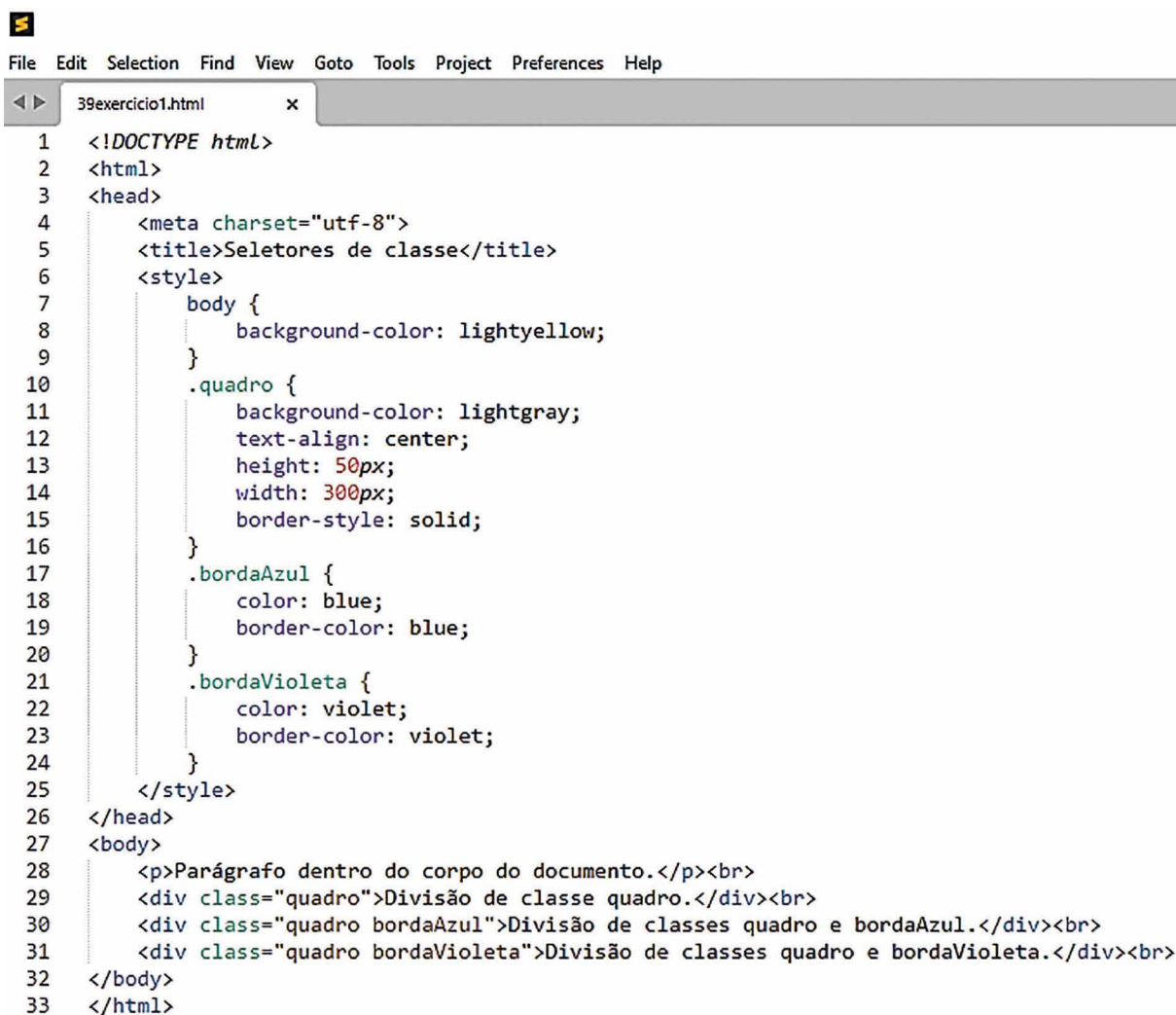


Figura 106 – Saída do exemplo com seletores de ID

5.4.1.3 Seletores de classe

Os seletores de classe utilizam o atributo `class` de elementos HTML para selecioná-los. Diferentemente do ID, o atributo `class` não precisa ser único para elementos específicos, o que faz com que uma mesma regra de estilo possa ser aplicada a diversos elementos em uma mesma página. Além disso, um mesmo elemento pode adotar mais de uma classe. Seletores de classe são amplamente utilizados em CSS, devido à sua versatilidade.

Para criarmos esse tipo de seletor, basta digitarmos o caractere de ponto final, seguido do nome que desejamos adotar para nossa classe. Acompanhe o exemplo a seguir.



```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Seletores de classe</title>
6      <style>
7          body {
8              background-color: lightyellow;
9          }
10         .quadro {
11             background-color: lightgray;
12             text-align: center;
13             height: 50px;
14             width: 300px;
15             border-style: solid;
16         }
17         .bordaAzul {
18             color: blue;
19             border-color: blue;
20         }
21         .bordaVioleta {
22             color: violet;
23             border-color: violet;
24         }
25     </style>
26 </head>
27 <body>
28     <p>Parágrafo dentro do corpo do documento.</p><br>
29     <div class="quadro">Divisão de classe quadro.</div><br>
30     <div class="quadro bordaAzul">Divisão de classes quadro e bordaAzul.</div><br>
31     <div class="quadro bordaVioleta">Divisão de classes quadro e bordaVioleta.</div><br>
32 </body>
33 </html>
    
```

Figura 107 – Exemplo de código com seletores de classe

Na folha de estilos do cabeçalho, temos um seletor direto `body`, que ajustará um fundo amarelo-claro para todo o corpo do nosso documento. Também foram criadas três classes, que atuam como seletores: `quadro`, `bordaAzul` e `bordaVioleta`.

Na classe `quadro`, indicamos as seguintes propriedades: fundo cinza-claro, com alinhamento centralizado do texto, altura de 50 px e largura de 300 px, com borda sólida. Na classe `bordaAzul`, indicamos que o texto e a borda devem ter a cor azul. Na classe `bordaVioleta`, mostramos que o texto e a borda devem ter a cor violeta.

No corpo do nosso documento HTML, encontramos um elemento `<p>`, que não possui uma classe especificada. Já os elementos `<div>` possuem classes: a primeira adota as propriedades da classe `quadro`; a segunda une as características das classes `quadro` e `bordaAzul`; e a terceira une as características das classes `quadro` e `bordaVioleta`. Esse exemplo evidencia bem a natureza em cascata do CSS. O resultado no navegador pode ser observado na figura a seguir.

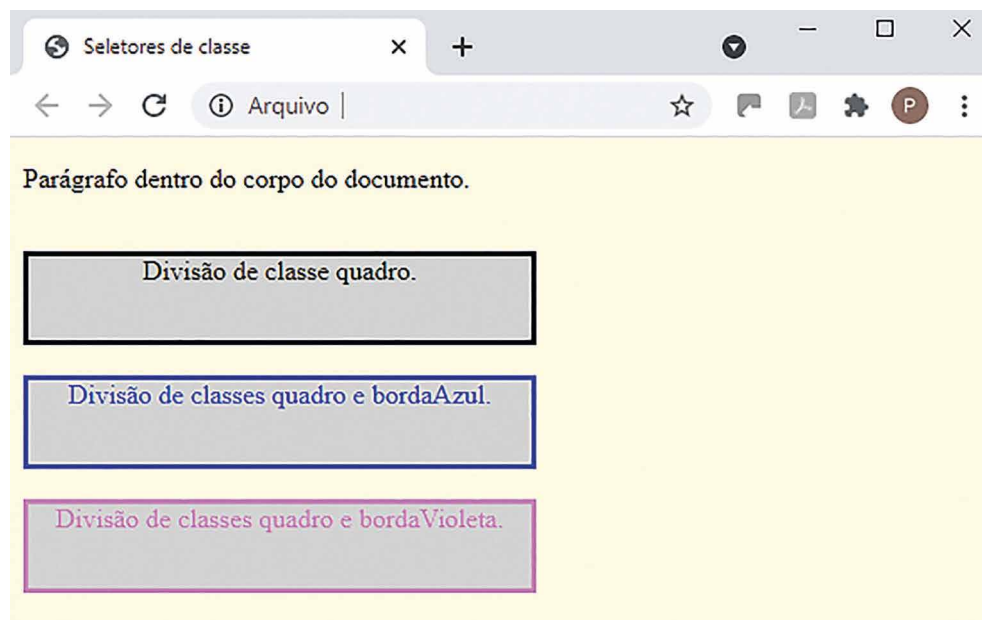


Figura 108 – Saída do exemplo com seletores de classe



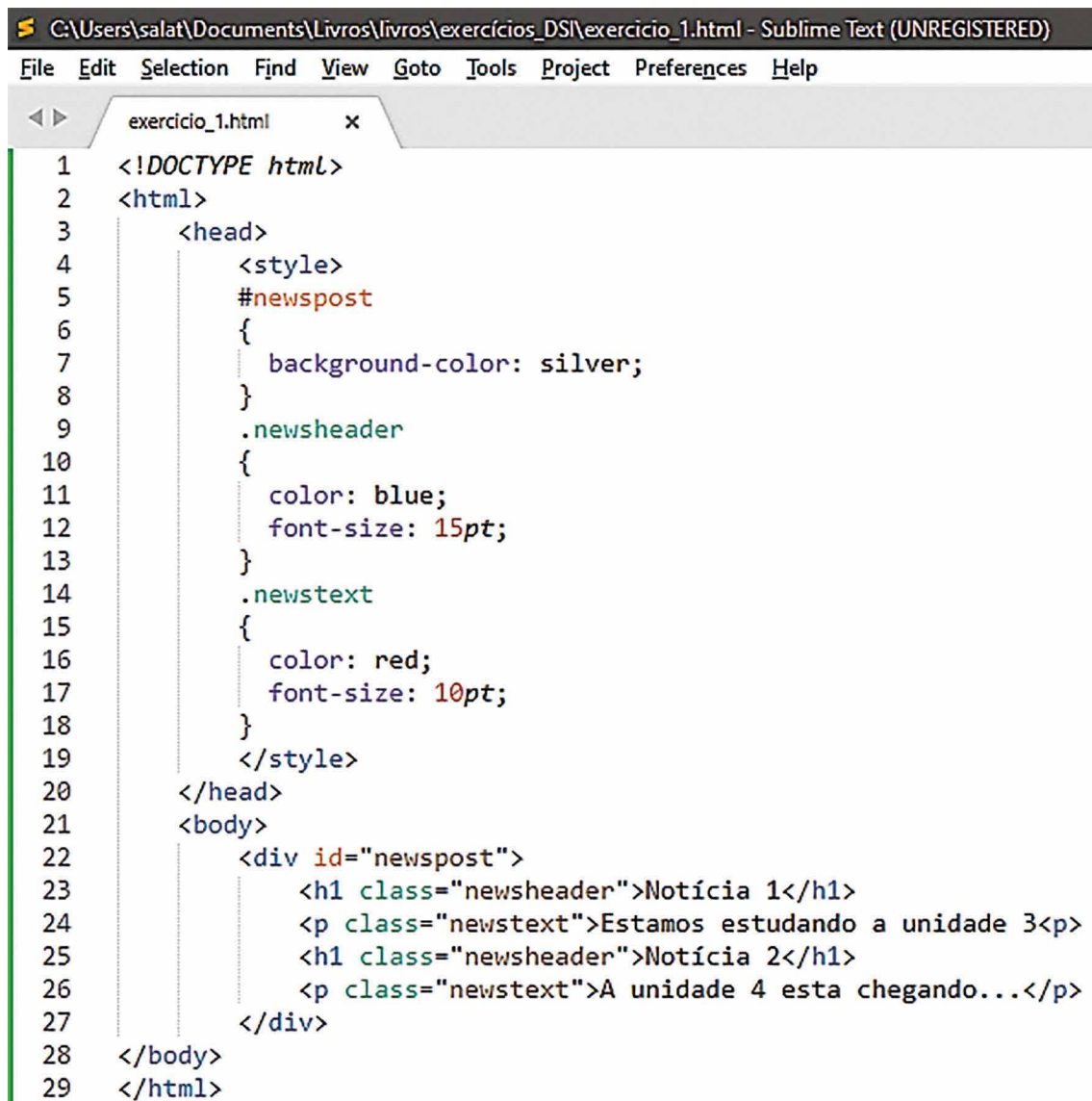
Lembrete

Todas as propriedades não especificadas na folha de estilos são adotadas de acordo com o padrão do navegador.

Exemplo de aplicação

Suponha que estamos desenvolvendo um documento HTML contendo postagens de notícias acadêmicas. Todas as publicações de notícias no documento precisam obedecer ao mesmo conjunto de regras de estilo e, dessa forma, assumem a mesma aparência para cada publicação de notícia.

Cada postagem pode ser exibida em um plano de fundo de cor diferente do restante da página e sempre deve ter cabeçalhos em negrito e em azul com texto de parágrafo menor em vermelho, conforme implementação realizada na figura.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       #newspost
6       {
7         background-color: silver;
8       }
9       .newsheader
10      {
11        color: blue;
12        font-size: 15pt;
13      }
14      .newstext
15      {
16        color: red;
17        font-size: 10pt;
18      }
19    </style>
20  </head>
21  <body>
22    <div id="newspost">
23      <h1 class="newsheader">Notícia 1</h1>
24      <p class="newstext">Estamos estudando a unidade 3</p>
25      <h1 class="newsheader">Notícia 2</h1>
26      <p class="newstext">A unidade 4 esta chegando...</p>
27    </div>
28  </body>
29 </html>
```

Figura 109 – Implementação de CSS

O código, que utiliza seletores de ID e de classe, produz a saída apresentada na figura a seguir. Desde que os elementos estejam dentro de uma seção que tenha o ID nomeado, eles serão coloridos de acordo.

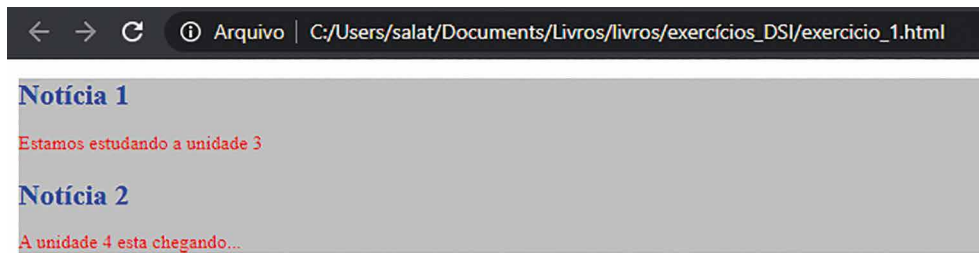


Figura 110 – Resultado da implementação de CSS

5.4.1.4 Seletor universal (*)

O seletor universal, identificado pelo símbolo * (asterisco), seleciona e afeta todos os elementos HTML de uma página. Dessa forma, ele se comporta como o seletor mais abrangente de todos.

No início, essa ação pode ser encarada com bons olhos, mas seu uso pode tornar nossa página mais lenta em um dispositivo móvel ou até mesmo transformá-la em uma página com muitos elementos. Por isso, o seletor universal deve ser utilizado com muita moderação.

O seletor universal torna possível a implementação de propriedades como parte de uma configuração inicial. Seu uso é bastante conhecido no reset universal do CSS, visto no código da figura a seguir, que define a exclusão das margens (na linha `margin: 0;`) e do preenchimento (na linha `padding: 0;`) de todos os elementos HTML da página. Esse reset pode ser utilizado para sobrescrever propriedades de estilo padrão de navegadores, apagando configurações de estilo nativas do browser e garantindo que sua página terá aparência similar em diferentes navegadores.



Figura 111 – Reset universal do CSS utilizando o seletor universal

Vamos, agora, aprender alguns detalhes a respeito das propriedades CSS utilizadas no bloco de declaração do nosso seletor universal. A propriedade `margin` simplesmente adiciona uma margem externa ao elemento. Podemos utilizar qualquer medida CSS (px, pt, em, %...) como tamanho da propriedade. Além disso, podemos atribuir valores negativos. Essa propriedade pode ser dividida em margens individuais, conforme exposto no quadro a seguir.

Quadro 11 – Propriedade `margin`

#elemento {
<code>margin-top: 15 px; /* – propriedade define a margem superior de um elemento</code>
<code>margin-right: 10 px; /* – propriedade define a margem direita de um elemento</code>
<code>margin-bottom: 25 px; /* – propriedade define a margem inferior de um elemento</code>
<code>margin-left: 35 px; /* – propriedade define a margem esquerda de um elemento</code>
}

A propriedade `padding`, que significa preenchimento, tem um funcionamento muito similar ao do `margin`, porém, em vez de dar um espaçamento externo, ele realiza o espaçamento interno da página. Essa propriedade também pode ser dividida em preenchimentos individuais, conforme exposto no quadro a seguir.

Quadro 12 – Propriedade `padding`

#elemento {
<code>padding-top: 25 px; /* – propriedade que define o espaço de preenchimento superior de um elemento</code>
<code>padding-right: 45 px; /* – propriedade que define o espaço de preenchimento no lado direito de um elemento</code>
<code>padding-bottom: 35 px; /* – propriedade que define o espaço de preenchimento inferior de um elemento</code>
<code>padding-left: 15 px; /* – propriedade que define o espaço de preenchimento no lado esquerdo de um elemento.</code>
}
<code>#elemento { padding: 25 px 15 px; } /* Top/bottom – right/left */</code>
<code>#elemento { padding: 25 px; } /* top/right/bottom/left */</code>

Deve-se ter cuidado ao utilizar o `padding`, pois ele aumenta a largura do elemento. Caso seu elemento tenha um `width` de 200 px e utilizarmos um `padding-left` de 50 px, ele vai passar a ter uma largura total de 250 px. Então, sempre que necessário, deve-se compensar o `padding` do elemento reduzindo o `width` ou o `height`.

Na figura a seguir, conseguimos visualizar a diferença entre as propriedades `margin` e `padding`. O tamanho do elemento em si é definido por suas bordas, representadas pelo retângulo vermelho. A margem é aplicada para fora do elemento e deve criar um espaçamento entre as bordas do elemento em questão e o restante dos elementos da página. Já o `padding` irá criar espaçamento entre as bordas do elemento e o seu próprio conteúdo interno, que ficará restrito ao retângulo preto menor da imagem.

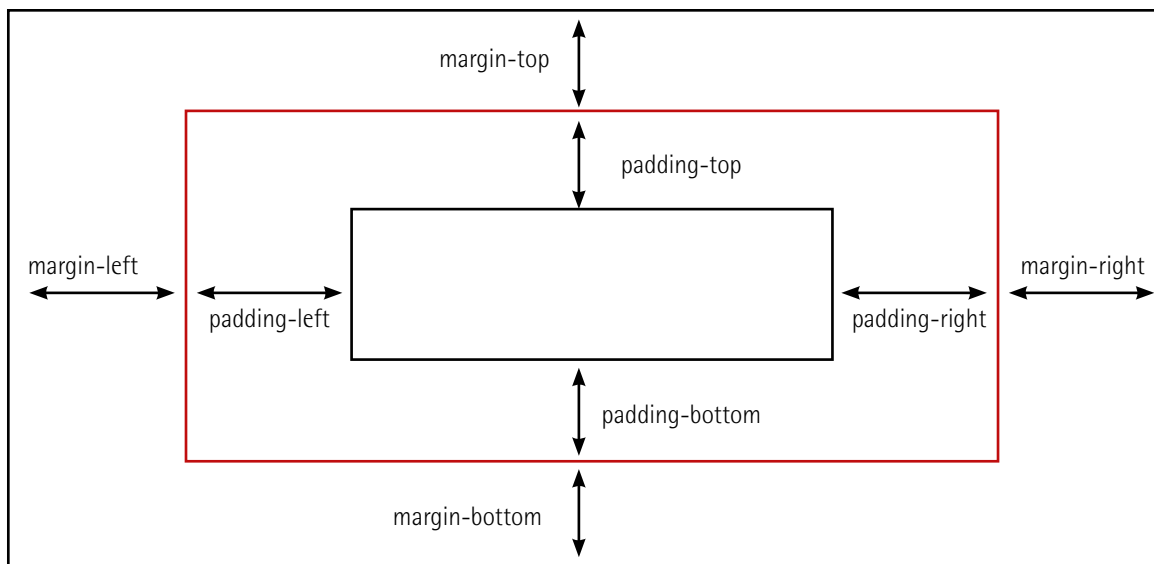


Figura 112 – Margin/padding – CSS

Depois que incluímos um *reset* universal em algum lugar próximo à parte superior da nossa folha de estilos, podemos ter certeza de que imediatamente nenhuma das suas regras terá margens ou preenchimentos aplicados a elas.



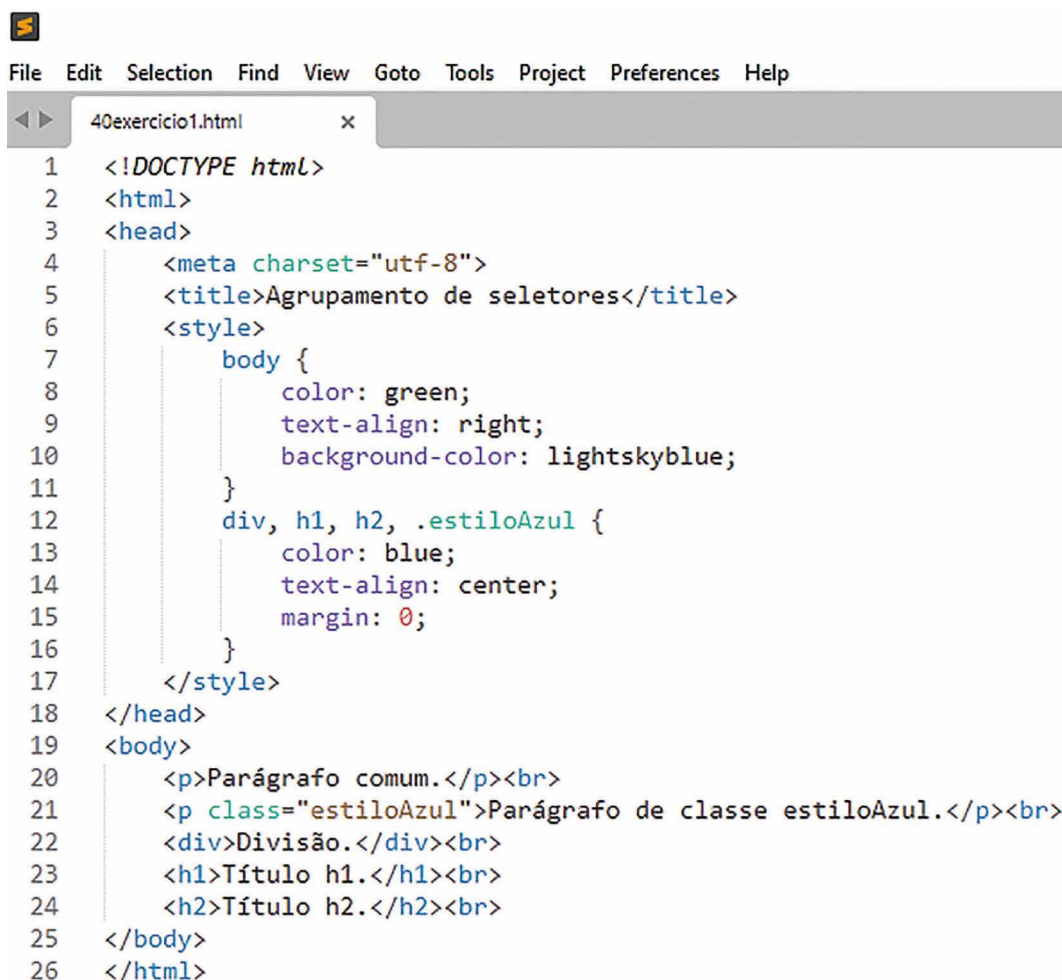
Lembrete

A propriedade `margin` simplesmente adiciona uma margem ao elemento. Pode-se utilizar qualquer medida CSS (px, pt, em, %...), como tamanho. Além disso, pode-se atribuir valores negativos.

O `padding` tem um funcionamento muito similar ao do `margin`, porém, em vez de dar um espaçamento externo, ele realiza o espaçamento interno da página.

5.4.1.5 Agrupamento de seletores

Podemos agrupar seletores simples que compartilham uma mesma regra de estilo. Dessa forma, nosso código se torna mais conciso e organizado. Para isso, podemos separar nossos seletores por vírgula, construindo apenas um bloco de declaração que servirá para todos eles. Acompanhe o exemplo da figura a seguir.



```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Agrupamento de seletores</title>
6      <style>
7          body {
8              color: green;
9              text-align: right;
10             background-color: lightskyblue;
11         }
12         div, h1, h2, .estiloAzul {
13             color: blue;
14             text-align: center;
15             margin: 0;
16         }
17     </style>
18 </head>
19 <body>
20     <p>Parágrafo comum.</p><br>
21     <p class="estiloAzul">Parágrafo de classe estiloAzul.</p><br>
22     <div>Divisão.</div><br>
23     <h1>Título h1.</h1><br>
24     <h2>Título h2.</h2><br>
25 </body>
26 </html>
    
```

Figura 113 – Exemplo de código com seletores agrupados

Note que, no código, separamos os seletores `div`, `h1`, `h2` e `estiloAzul` por vírgulas. Dessa forma, o bloco de declaração de todos esses seletores será o mesmo. O primeiro elemento `<p>` do corpo do nosso documento é afetado apenas pela declaração do seletor `body`. Já o segundo, que possui a classe `estiloAzul`, adota as propriedades desse seletor. Como resultado, temos o que se mostra na figura seguinte.



Figura 114 – Saída do exemplo com seletores agrupados

5.4.2 Seletores combinados

Os seletores combinados utilizam combinadores em sua estrutura. Esses combinadores estabelecem uma relação entre dois ou mais seletores simples, compondo um seletor combinado.

Existem quatro combinadores no CSS, indicados a seguir.

- Descendente (espaço).
- Filho direto (>).
- Irmão adjacente (+).
- Irmão geral (~).

Vamos estudá-los a seguir.

5.4.2.1 Descendente (espaço)

O combinador descendente é composto por um espaço entre seletores simples. Por meio dele, selecionamos apenas elementos específicos que estão contidos dentro de outros. Observe o exemplo da figura a seguir.

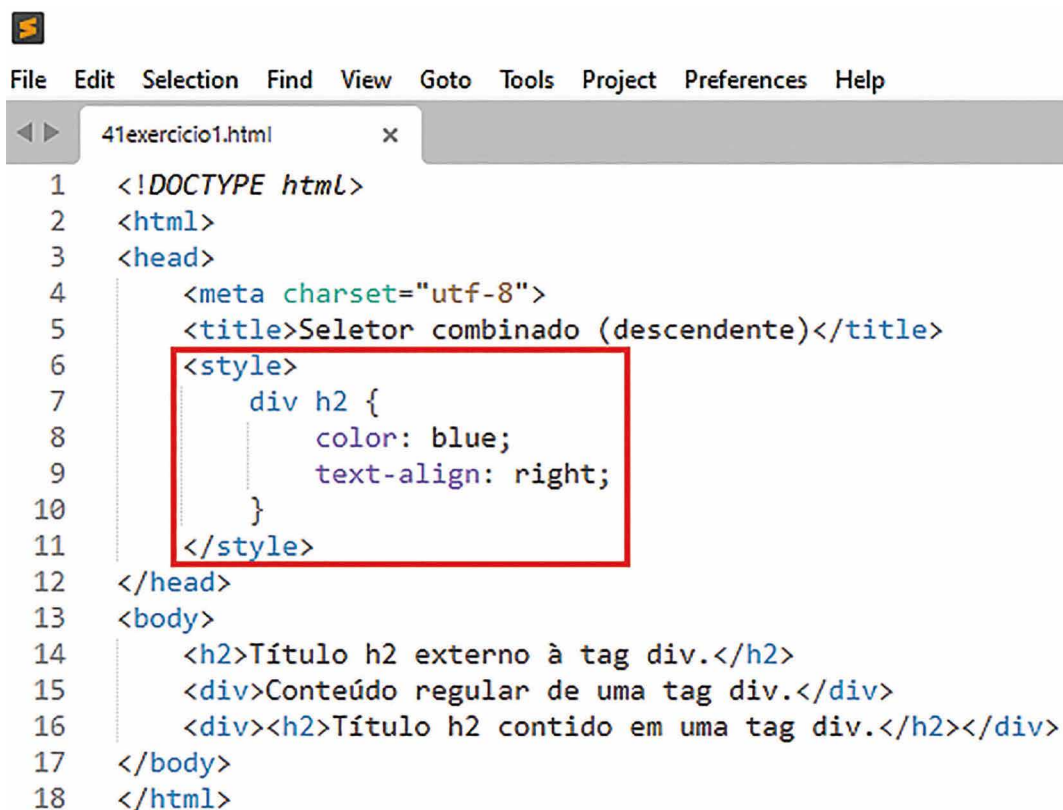


Figura 115 – Exemplo de código com combinador descendente

O seletor combinado `div h2` irá selecionar todos os elementos `<h2>` contidos dentro de um elemento `<div>`. Dessa forma, ele não afetará qualquer outro conteúdo de um `<div>`, nem títulos `<h2>` que não integram um `<div>`. Apenas o conteúdo da linha 16 do código será exibido com as propriedades do seletor combinado. O resultado pode ser observado na figura seguinte.



Figura 116 – Saída do exemplo com combinador descendente

5.4.2.2 Filho direto (>)

O combinador filho direto proporciona a seleção de um elemento diretamente contido em outro. Para utilizá-lo, basta colocar o símbolo `>` entre seletores simples. Avalie o exemplo a seguir.

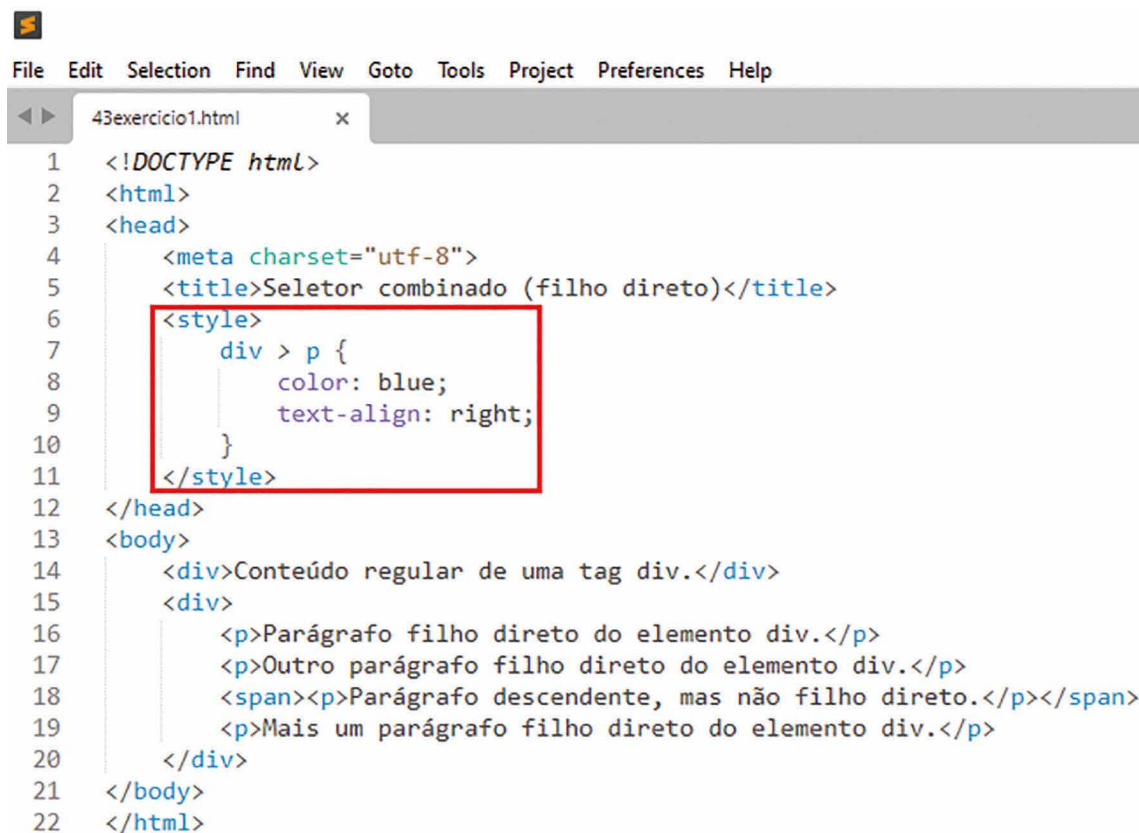


Figura 117 – Exemplo de código com combinador filho direto

No exemplo, apenas os elementos `<p>` que estão diretamente contidos em elementos `<div>` serão afetados pelo seletor combinado `div > p`. Por "diretamente contidos", queremos dizer que não há nenhum elemento intermediário entre o elemento "pai" e o elemento "filho". Se observarmos a linha 18 do código, vemos que a hierarquia entre elementos é a seguinte: `div > span > p`. Dessa forma, o elemento `<p>` da linha 18 não é filho direto do elemento `<div>` que o contém, pois o elemento `` encontra-se entre eles. O resultado pode ser observado na figura seguinte.

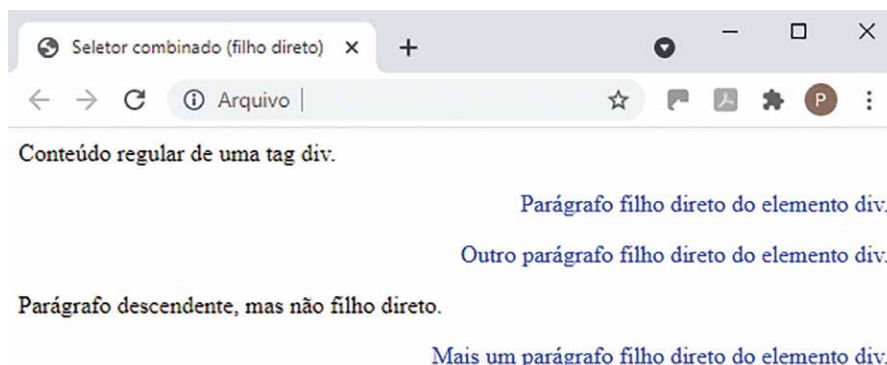


Figura 118 – Saída do exemplo com combinador filho direto

5.4.2.3 Irmão adjacente (+)

O combinador irmão adjacente proporciona a seleção de um elemento imediatamente posterior a outro. Para utilizá-lo, basta colocar o símbolo `+` entre seletores simples. Acompanhe o exemplo a seguir.



Figura 119 – Exemplo de regra CSS para demonstrar o combinador irmão adjacente

No caso, apenas os elementos `<p>` que se encontram imediatamente após um elemento `<h1>` serão selecionados. Dessa forma, o conteúdo das linhas 14 e 17 será afetado pelas propriedades do seletor combinado `h1 + p`, mas todos os outros elementos não serão. Assim, obtém-se o resultado a seguir.



Figura 120 – Resultado da aplicação do combinador adjacente no seletor

5.4.2.4 Irmão geral (~)

O combinador irmão geral funciona de forma similar ao irmão adjacente, mas não exige que o elemento a ser selecionado seja imediatamente posterior a outro. Para utilizá-lo, basta colocar o símbolo `~` entre seletores simples. Acompanhe o exemplo a seguir.



Figura 121 – Exemplo de código com combinador irmão geral

No exemplo, temos o seletor combinado `div ~ p`, que deverá selecionar todos os elementos `<p>` posicionados após um `<div>`. Dessa forma, ele seleciona os elementos das linhas 18 e 20 do código. Note que não há necessidade de adjacência, o que torna o combinador irmão geral mais abrangente do que o combinador irmão adjacente. O resultado pode ser visto na figura a seguir.

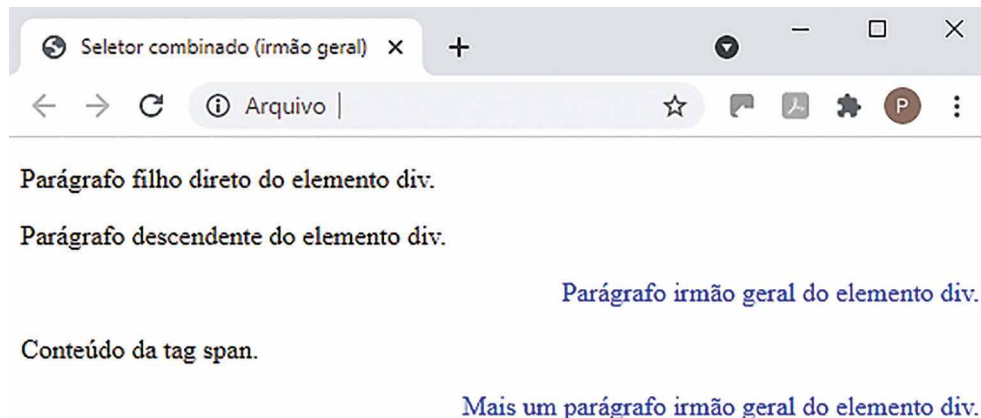


Figura 122 – Saída do exemplo com combinador irmão geral

5.4.3 Seletores de atributo

No CSS3, os seletores de atributos são extremamente poderosos e possibilitam que montemos um estilo bastante inteligente dos elementos. Os seletores de atributos podem ser definidos como um conjunto personalizado de expressões de pesquisa, especificamente criado para identificar elementos com base no valor ou em parte de um atributo. Para começarmos o nosso estudo, considere o código apresentado na figura.

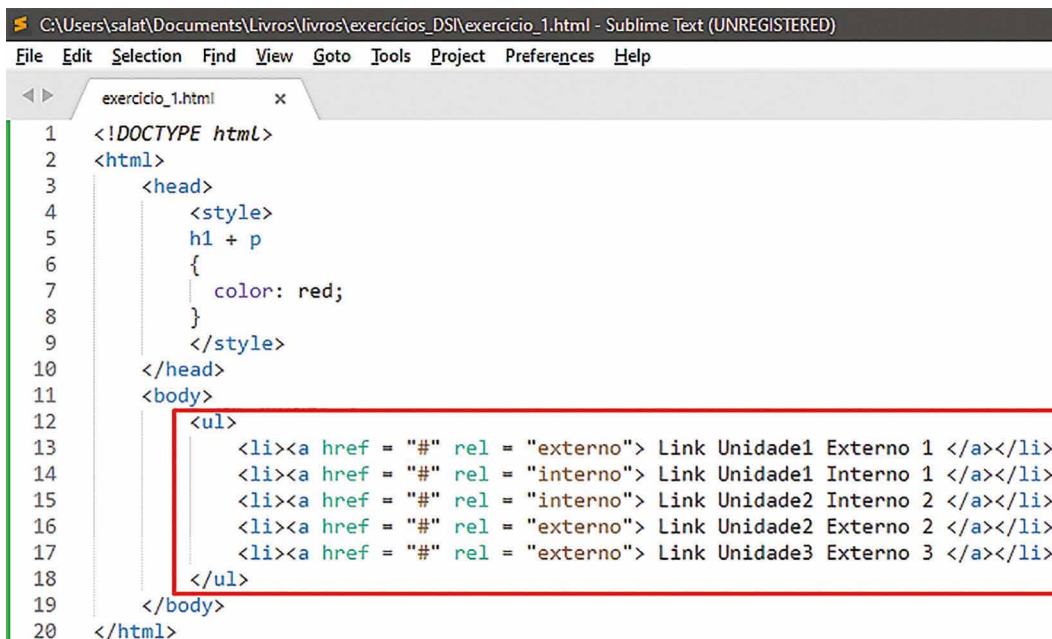
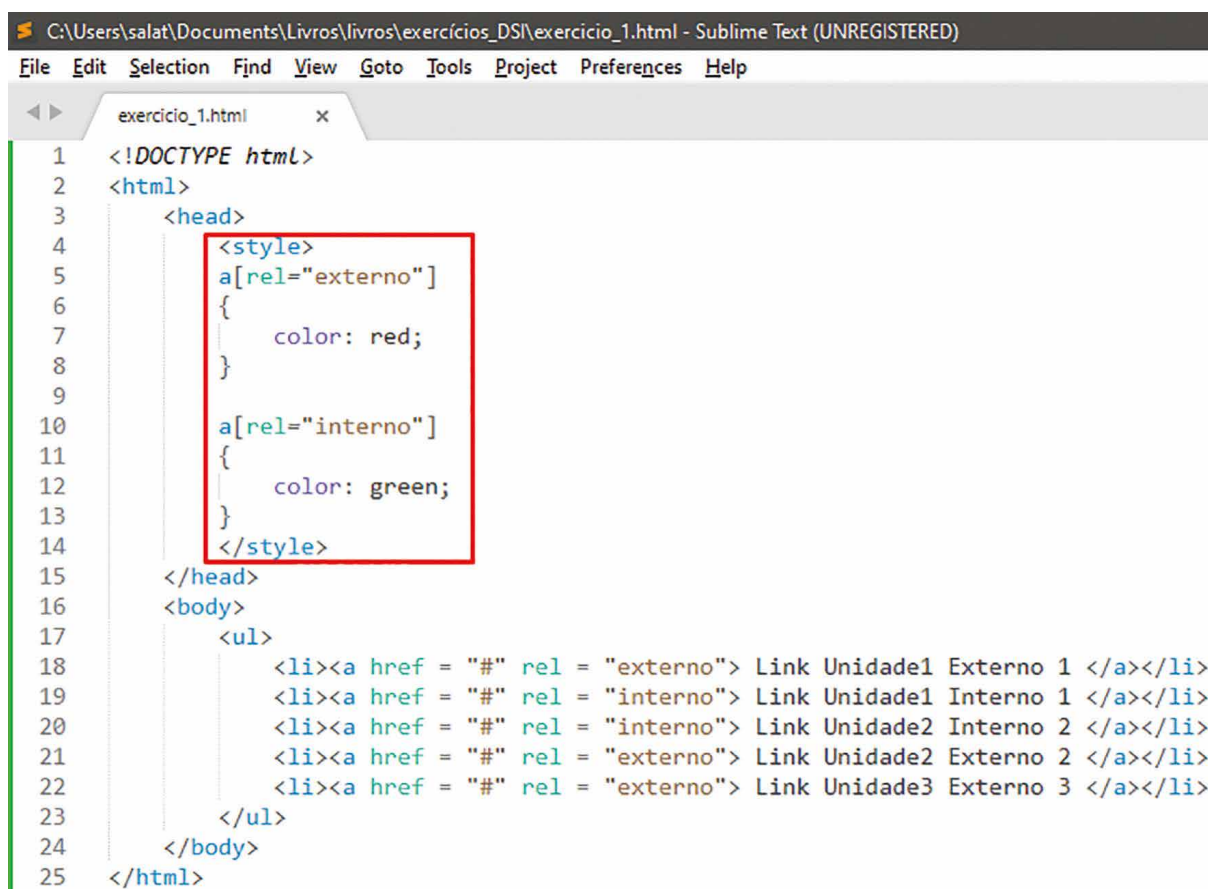


Figura 123 – Exemplo de marcação HTML com atributos personalizados

É possível ver, no exemplo da figura, que criamos uma lista não ordenada, cujos itens são âncoras HTML (elementos do tipo `<a>`). Note que utilizamos o atributo **rel** para especificar se o link de cada item é externo ou interno.

Se quisermos estilizar cada um desses tipos de maneiras diferentes, podemos apenas adicionar uma classe, como `.internal` ou `.external`. No entanto, pode haver situações em que a marcação HTML é gerada dentro de um processo sobre o qual não se tenha esse controle; assim, talvez não seja possível adicionar classes de saída dessa maneira. Para resolver essa situação, podemos utilizar o seletor de atributos para buscar um atributo específico dentro do elemento desejado.

Primeiramente, vamos observar as figuras a seguir, que mostram como podemos aplicar seletores de atributo aos links vistos anteriormente. Vamos adicionar regras de estilo à nossa folha de estilos, supondo que já fizemos a inclusão do HTML anterior ao nosso documento.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       a[rel="externo"]
6       {
7         color: red;
8       }
9
10      a[rel="interno"]
11      {
12        color: green;
13      }
14    </style>
15  </head>
16  <body>
17    <ul>
18      <li><a href = "#" rel = "externo"> Link Unidade1 Externo 1 </a></li>
19      <li><a href = "#" rel = "interno"> Link Unidade1 Interno 1 </a></li>
20      <li><a href = "#" rel = "interno"> Link Unidade2 Interno 2 </a></li>
21      <li><a href = "#" rel = "externo"> Link Unidade2 Externo 2 </a></li>
22      <li><a href = "#" rel = "externo"> Link Unidade3 Externo 3 </a></li>
23    </ul>
24  </body>
25 </html>
```

Figura 124 – Exemplo de regras com seletores de atributos

Na linha 5, vemos o código `a[rel="externo"]`, que representa um seletor de atributo. O que ele faz é procurar elementos `<a>` cujo atributo `rel` possua o valor `externo`. Para os elementos que atendam a essa especificação, o texto deve aparecer em vermelho na execução. Perceba que, entre colchetes, especificamos qual atributo do elemento `<a>` devemos buscar para que o bloco de declaração do seletor seja executado.

Já na linha 10, vemos `a[rel="interno"]`, que é outro seletor de atributo. Nesse caso, o bloco de declaração será executado para elementos `<a>` cujo atributo `rel` possua o valor `interno`, fazendo com que o texto seja exibido em verde.

Utilizando apenas dois seletores distintos, segmentamos de maneira fácil os dois tipos diferentes de links, simplesmente por causa da diferença no atributo `rel`. A principal vantagem do estudo realizado é que agora não precisamos nos preocupar em garantir que uma classe específica seja aplicada a determinado link, tornando a manutenção do documento muito mais fácil e ágil e menos propensa a erros. A execução do código no navegador é mostrada na figura a seguir.

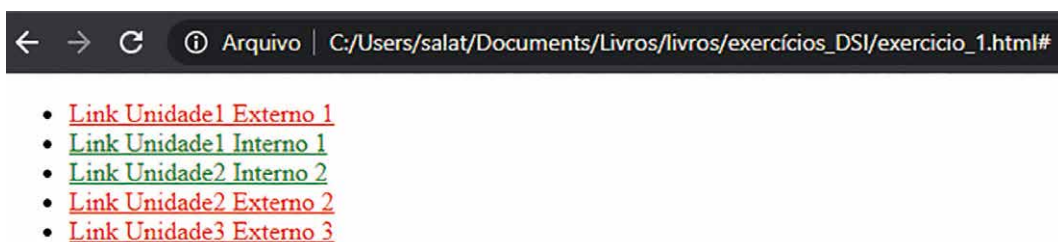


Figura 125 – Resultado da aplicação de regras com seletores de atributos

Utilizamos o atributo `rel` para definir um relacionamento interno ou um relacionamento externo em um elemento âncora. Em termos de especificações do W3C, essa provavelmente não seria a melhor forma de fazer isso, e o HTML poderia falhar na validação caso fosse testado com uma ferramenta de validação do HTML5. Contudo, isso não significa que precisamos voltar a usar um seletor de classe.

Exemplo de aplicação

Vamos imaginar que todos os seus links externos tenham uma URL totalmente qualificada, e todos os seus links internos tenham apenas uma folha de página relativa. Nesse caso, poderíamos escrever a marcação da maneira apresentada a seguir.

```
<ul>
  <li><a href="http://externo.com/link1">Link Externo 1</a></li>
  <li><a href="/link1">Link Interno 1</a></li>
  <li><a href="/link2">Link Interno 2</a></li>
  <li><a href="http://externo.com/link2">Link Externo 2</a></li>
  <li><a href="http://externo.com/link3">Link Externo 3</a></li>
</ul>
```

Figura 126 – Lista HTML com links externos e internos

Se examinarmos a diferença nos atributos `href`, podemos perceber que todos os links externos começam com `http://` e os internos não. Nessa situação, podemos utilizar um seletor de atributo cuja regra de busca é "o valor do atributo começa com", colocando o símbolo `^` após o nome do atributo. Agora podemos estilizar facilmente os externos com uma cor diferente. Observe o código da figura a seguir.

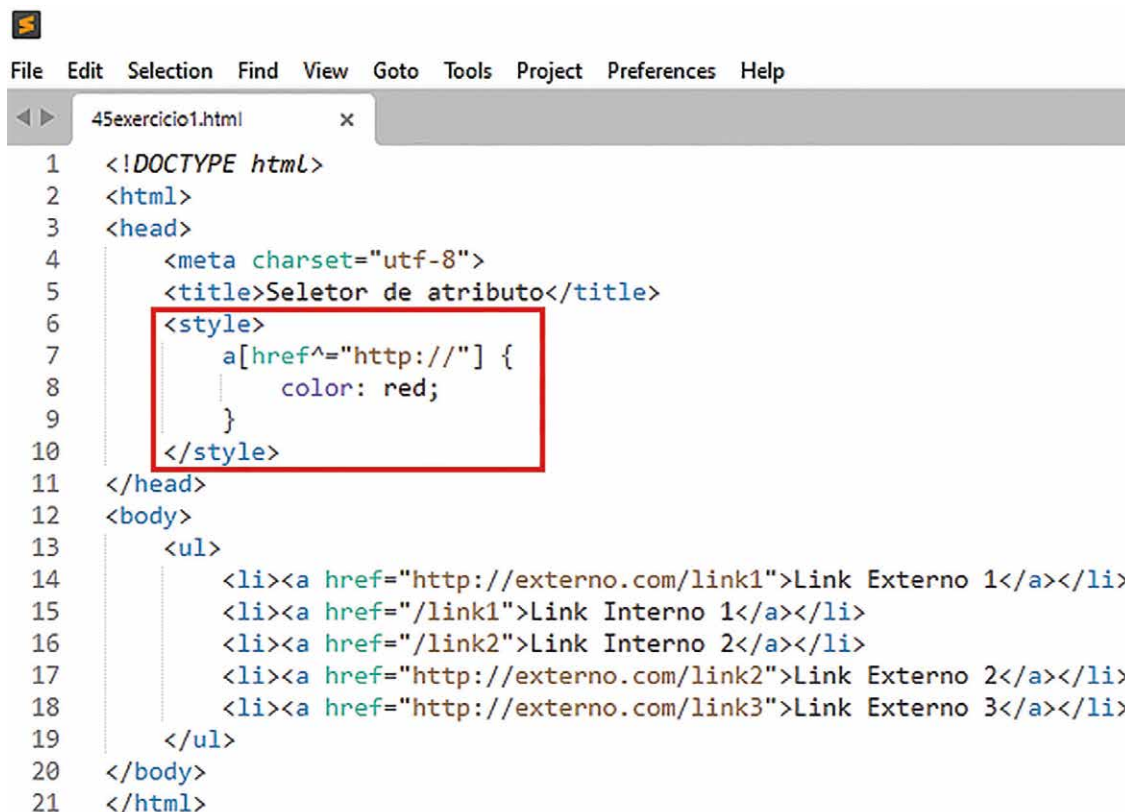


Figura 127 – Código com seletor de atributo com regra de busca especial

Nosso seletor de atributo `a[href^="http://"]` selecionará os elementos `<a>` cujo valor do atributo `href` começa com a sequência de caracteres `http://`, aplicando cor vermelha ao texto. Os elementos `<a>` que não se encaixam no critério desse seletor assumirão a regra de estilo padrão do navegador. Dessa forma, a execução do código nos leva ao resultado apresentado a seguir.



Figura 128 – Resultado da aplicação do seletor de atributo com regra de busca especial

O que abordamos aqui nem sequer arranhou a superfície do potencial dos seletores de atributos. No HTML5, há um novo conceito de atributo chamado de atributo de dados (*data attribute*), em que podemos personalizar os nomes desses atributos, desde que iniciem pelos caracteres `data-`. Por exemplo, podemos criar os seguintes atributos customizados para nossos elementos HTML:

- data-username;
- data-recordindex.

É possível utilizar esses atributos de dados em qualquer elemento, para qualquer finalidade. Quando efetuamos a combinação com os seletores de atributo corretos, eles permitem utilizar o HTML5 para descrever facilmente suas próprias intenções.



Saiba mais

Para conhecer mais a respeito de atributos de dados em HTML5, leia o artigo indicado a seguir.

DEVMEDIA. *Utilizando os Custom Data Attributes da HTML5*. 2013. Disponível em: <https://bit.ly/3FvCOTE>. Acesso em: 10 maio 2022.

5.4.4 Seletores de pseudoclasses

No CSS, uma pseudoclassee é uma palavra-chave posicionada após o nome de um seletor, que especifica um estado do elemento selecionado. Com uma pseudoclassee, podemos alterar uma regra de estilo mediante a ocorrência de um evento externo. Um exemplo muito conhecido disso é a pseudoclassee :hover, bastante utilizada nas tags de âncora HTML e que permite a mudança da cor do texto quando o mouse do usuário está posicionado em cima do elemento durante a execução no navegador.

Para implementar seletores desse tipo, basta escrever o nome do seletor, o símbolo : e o nome da pseudoclassee desejada. Temos, portanto, a sintaxe geral para uma regra de estilo que utiliza um seletor de pseudoclassee mostrada a seguir.

```
seletor:pseudoclassee {  
  propriedade="valor"  
}
```

Vamos estudar, na sequência, a implementação de alguns tipos de seletores de pseudoclassee de acordo com as regras do CSS3.

5.4.4.1 Pseudoclasses utilizadas com o elemento <a>

As âncoras (elementos <a>) são um tipo de elemento HTML em que é muito comum utilizarmos seletores atrelados a pseudoclasses. Para isso, costuma-se utilizar as pseudoclasses mostradas a seguir.

- **:link**: estiliza um elemento <a> que ainda não foi visitado.
- **:visited**: estiliza um elemento <a> que já foi visitado.

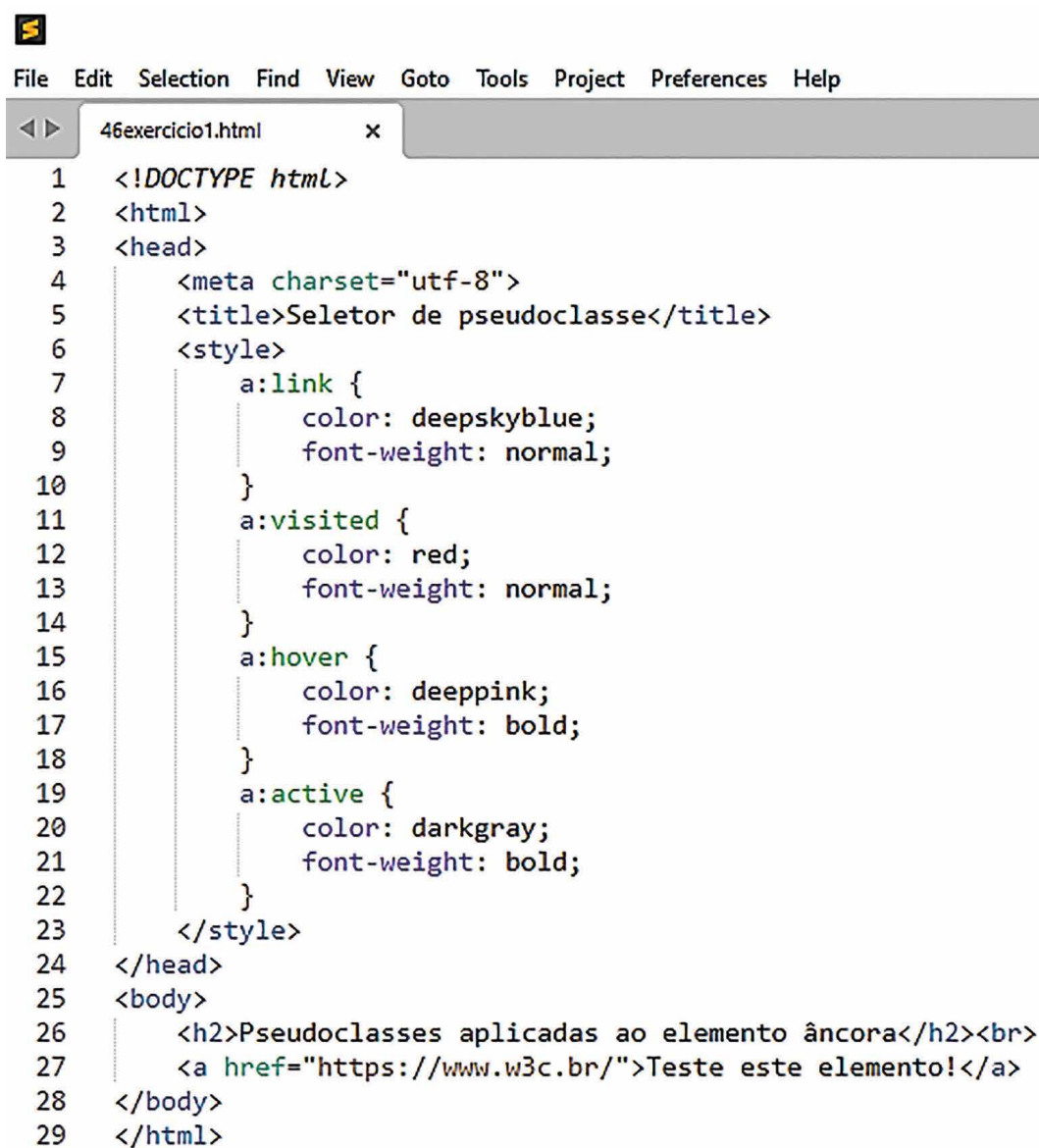
- **:hover**: estiliza o elemento sobre o qual o cursor do mouse do usuário está posicionado.
- **:active**: estiliza o elemento selecionado (ativo) no momento.



Observação

As pseudoclasses **:hover** e **:active** podem ser utilizadas com qualquer elemento HTML, não apenas com o elemento **<a>**.

Observe o código da figura a seguir, em que estilizamos uma âncora HTML com seletores de pseudoclasses.



```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Seletor de pseudoclasse</title>
6      <style>
7          a:link {
8              color: deepskyblue;
9              font-weight: normal;
10         }
11         a:visited {
12             color: red;
13             font-weight: normal;
14         }
15         a:hover {
16             color: deeppink;
17             font-weight: bold;
18         }
19         a:active {
20             color: darkgray;
21             font-weight: bold;
22         }
23     </style>
24 </head>
25 <body>
26     <h2>Pseudoclasses aplicadas ao elemento âncora</h2><br>
27     <a href="https://www.w3c.br/">Teste este elemento!</a>
28 </body>
29 </html>
```

Figura 129 – Exemplo de código que utiliza seletores de pseudoclasses com elemento **<a>**

Na execução, verificamos que, a princípio, o texto "Teste este elemento!" está sendo exibido em azul. Ao passar o cursor do mouse sobre o elemento, ele deve ficar na cor rosa e em negrito. Ao clicar, o texto fica cinza e em negrito. Finalmente, ao visitarmos o endereço associado à âncora, o navegador deve passar a exibir o texto em vermelho (clique no link e, na sequência, acione o comando do seu navegador para voltar à página anterior).

Para testar esse código, abra o documento HTML no seu navegador, mas não no modo de navegação anônima. Caso contrário, o texto nunca será exibido em vermelho, pois não haverá histórico determinando que o endereço já foi visitado. Se você limpar o histórico de navegação após o endereço do W3C Brasil ser visitado, o texto "Teste este elemento!" deve voltar a ser exibido em azul.



Observação

Na folha de estilos CSS, o seletor `a:hover` precisa aparecer após os seletores `a:link` e `a:visited` para que sua ação funcione corretamente. Da mesma forma, o seletor `a:active` precisa aparecer após o seletor `a:hover`.

5.4.4.2 Pseudoclasses utilizadas com elementos de formulários

Os elementos de entrada de dados (como `<input>` ou `<option>`) também costumam se beneficiar de seletores de pseudoclasses. Portanto, as pseudoclasses que estudaremos a seguir são aplicáveis para estilizar formulários HTML.

Pseudoclas de foco (`:focus`)

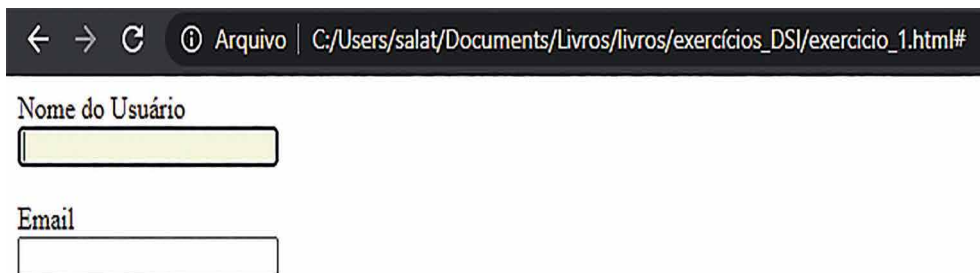
A pseudoclas `:focus` é utilizada quando um elemento de entrada, como o `<input>`, tem foco em um documento. Isso geralmente é utilizado para destacar um campo ativo de um formulário. A pseudoclas `:focus` pode ser utilizada em qualquer elemento que receba eventos do teclado ou outras entradas de dados vindas do usuário. Acompanhe o exemplo a seguir.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       :focus
6       {
7         background-color: beige;
8         color: green;
9       }
10    </style>
11  </head>
12  <body>
13    <form action="#" method="post">
14      <label for="txtUserName">Nome do Usuário</label><br />
15      <input type="text" id="txtUserName" name="txtUserName" /><br /><br />
16      <label for="txtEmail">Email</label><br />
17      <input type="text" id="txtEmail" name="txtEmail" /><br /><br />
18      <input type="submit" value="Submit" />
19    </form>
20  </body>
21 </html>
```

Figura 130 – Exemplo de código que aplica a pseudoclasse :focus

Para o nosso exemplo, qualquer elemento que receba o foco do usuário terá sua cor de fundo alterada. Na figura a seguir, o primeiro campo de entrada, que deve receber o nome do usuário, está em foco e aparece com fundo bege.



Nome do Usuário

Email

Figura 131 – Resultado do exemplo que aplica a pseudoclasse :focus

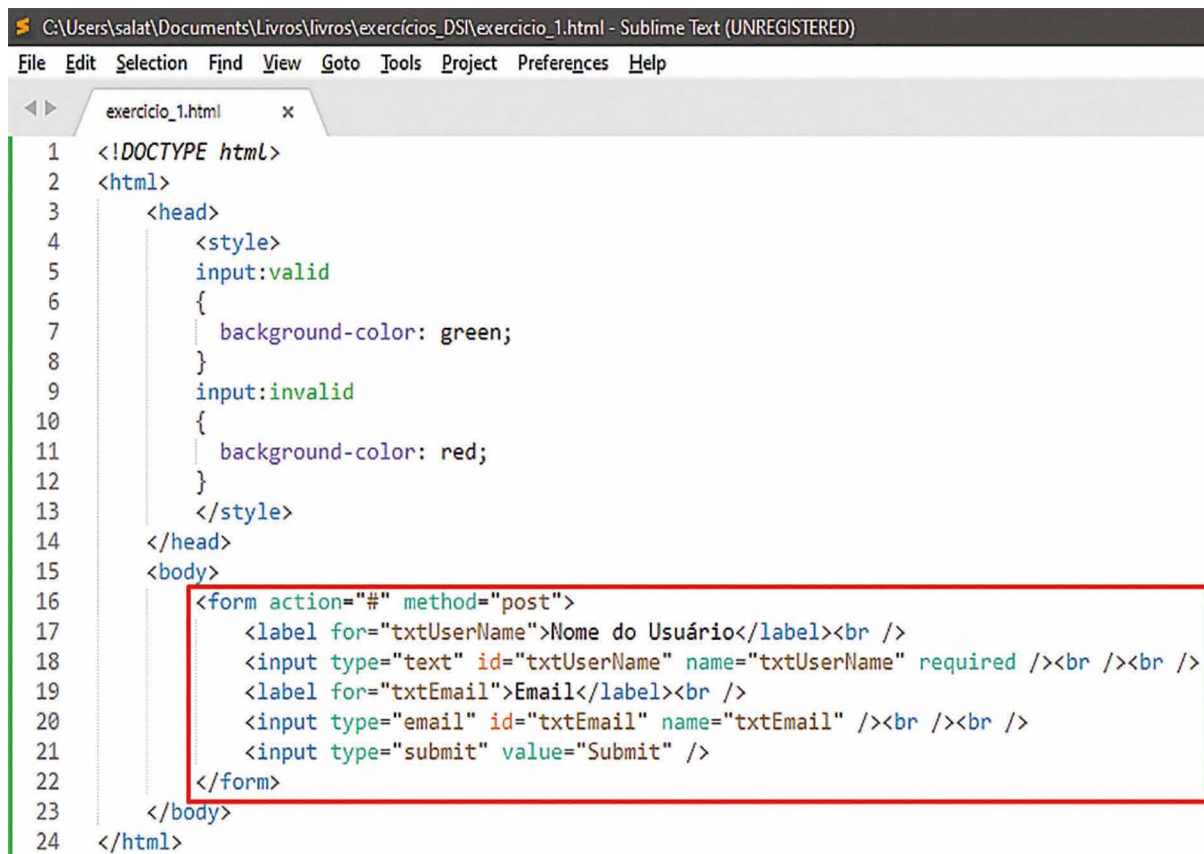
É importante ressaltar que consideramos que um elemento está em foco quando o cursor está piscando no input selecionado. No nosso exemplo, poderia ser no campo "nome do usuário" ou no campo "e-mail".

Pseudoclasses :valid e :invalid

Um dos ótimos recursos incrementados ao HTML5 é a possibilidade de validação do lado do browser, que pode ser utilizada para fazer com que o próprio browser efetue uma validação das entradas de um formulário antes de enviá-lo.

O seletor `:valid` seleciona elementos de um formulário que possuem um valor que esteja de acordo com o tipo de dado esperado, segundo as especificações dadas pelo programador para o elemento. Por sua vez, o seletor `:invalid` seleciona os elementos cujos valores de entrada não correspondem aos valores esperados.

Veja o código apresentado na figura a seguir.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       input:valid
6       {
7         background-color: green;
8       }
9       input:invalid
10      {
11        background-color: red;
12      }
13    </style>
14  </head>
15  <body>
16    <form action="#" method="post">
17      <label for="txtUserName">Nome do Usuário</label><br />
18      <input type="text" id="txtUserName" name="txtUserName" required /><br /><br />
19      <label for="txtEmail">Email</label><br />
20      <input type="email" id="txtEmail" name="txtEmail" /><br /><br />
21      <input type="submit" value="Submit" />
22    </form>
23  </body>
24 </html>
```

Figura 132 – Exemplo de marcação simulada de formulário HTML com elementos válidos

Se observarmos a caixa que recebe o nome de usuário (linha 18), poderemos visualizar que ela possui um atributo indicando que esse campo é obrigatório (*required*). Já na entrada de e-mail (linha 20), em vez de o tipo de entrada ser do tipo texto comum, podemos observar o trecho `type="email"`, que fará com que o navegador verifique se o usuário digitou um texto que corresponde à sintaxe de um endereço de e-mail. Esses elementos de validação devem ser correspondidos para que os campos sejam considerados válidos.

Quando observamos os seletores posicionados na folha de estilos, destacados na figura a seguir, percebemos que os campos considerados válidos devem aparecer com fundo verde. Já os campos inválidos são exibidos com fundo vermelho.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       input:valid
6       {
7         background-color: green;
8       }
9       input:invalid
10      {
11        background-color: red;
12      }
13    </style>
14  </head>
15  <body>
16    <form action="#" method="post">
17      <label for="txtUserName">Nome do Usuário</label><br />
18      <input type="text" id="txtUserName" name="txtUserName" required /><br />
19      <label for="txtEmail">Email</label><br />
20      <input type="email" id="txtEmail" name="txtEmail" /><br />
21      <input type="submit" value="Submit" />
22    </form>
23  </body>
24 </html>

```

Figura 133 – Exemplo de regras de validação CSS3

Ao executar o arquivo HTML no navegador, conforme apresentado na figura a seguir, o nome do usuário aparece imediatamente em vermelho, porque é um campo obrigatório. Dessa forma, até ser preenchido, será classificado como inválido.

O e-mail aparece inicialmente como válido, pois apenas exigimos que ele tenha sua entrada validada como um endereço de e-mail, mas não afirmamos que se trata de uma entrada obrigatória. Portanto, ter um e-mail em branco pode ser considerado válido. No entanto, se iniciarmos a nossa digitação, ele mudará para vermelho e voltará a verde até que a entrada tenha a sintaxe esperada ou seja removida por completo.

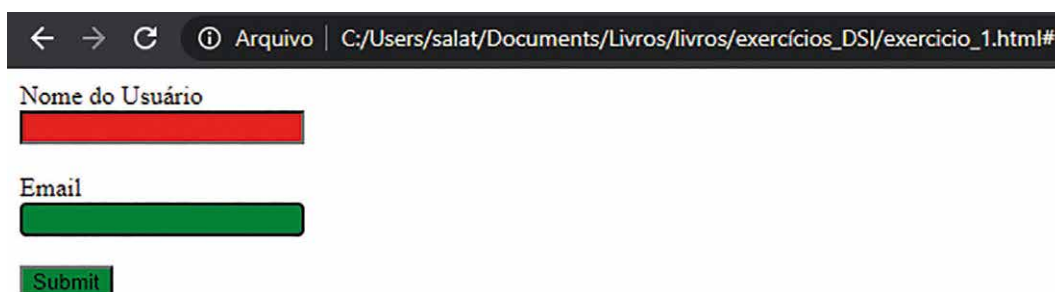


Figura 134 – Resultado do exemplo de regras de validação CSS3

Utilizar válido e inválido é uma excelente maneira de fornecer feedback ao usuário, mas vê-los imediatamente assim que o formulário é apresentado geralmente não é considerada uma boa prática.



Observação

Feedback pode ser definido como a informação que o emissor obtém da reação do receptor à sua mensagem, ou seja, tem por finalidade servir para avaliar os resultados da transmissão.

Logo, em geral, não os utilizamos diretamente aos elementos de entrada. Frequentemente anexamos a um nome de classe/ID e, em consequência, verificamos, aplicamos e utilizamos JavaScript quando o formulário é enviado.

Pseudoclasses :enabled e :disabled

Entre os muitos dos novos atributos que podem facilmente ser aplicados aos elementos de formulários na especificação do HTML5, está o atributo desativado (*disabled*). Podemos utilizá-lo como qualquer outro atributo simplesmente adicionando *disabled* a um elemento em sua marcação, conforme apresentado na figura a seguir, na linha 15. Se removermos o atributo *disabled* do elemento, a entrada retornará ao seu estado original (será considerada habilitada).

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       input:disabled
6       {
7         background-color: red;
8         border: 2px solid blue;
9       }
10    </style>
11  </head>
12  <body>
13    <form action="#" method="post">
14      <label for="txtUserName">User Name</label><br />
15      <input type="text" id="txtUserName" name="txtUserName" disabled /><br /><br />
16      <label for="txtEmail">Email</label><br />
17      <input type="text" id="txtEmail" name="txtEmail" /><br /><br />
18      <input type="submit" value="Submit" />
19    </form>
20  </body>
21 </html>
```

Figura 135 – Exemplo de formulário HTML com elementos desabilitados

Na sequência, adicionaremos uma regra simples à nossa folha de estilo, com a pseudoclasse *:disabled*, que verificará se o elemento `<input>` está desabilitado. Em caso positivo, será aplicado o bloco de declaração do seletor. Observe o código destacado na figura a seguir.

```

1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       input:disabled
6     {
7       background-color: red;
8       border: 2px solid blue;
9     }
10    </style>
11  </head>
12  <body>
13    <form action="#" method="post">
14      <label for="txtUserName">User Name</label><br />
15      <input type="text" id="txtUserName" name="txtUserName" disabled /><br /><br />
16      <label for="txtEmail">Email</label><br />
17      <input type="text" id="txtEmail" name="txtEmail" /><br /><br />
18      <input type="submit" value="Submit" />
19    </form>
20  </body>
21 </html>

```

Figura 136 – Exemplo de aplicação de regra CSS para estilizar elementos desabilitados

Essa regra geralmente é utilizada para apresentar campos somente de leitura em um editor de registros ou até mesmo para mostrar campos que não são aplicáveis ao formulário atual. Por exemplo, poderíamos realizar uma marcação de um formulário para que os campos de endereço pudessem ser visualizados apenas quando uma caixa de seleção para enviar e-mail para esse endereço estivesse marcada.

Se utilizássemos a estrutura da interface do usuário do *bootstrap*, poderíamos ver que alguns elementos, quando utilizados em um formulário, têm um pequeno círculo vermelho de parada apresentado sobre eles quando se passa o mouse com um ponteiro. Isso é realizado da mesma maneira utilizando a regra desativada para alterar o estilo do ponteiro do mouse.



Observação

Bootstrap pode ser definido como um *framework web* com código-fonte aberto para fins de desenvolvimento de componentes de interface e *front-end* para sites e de aplicações web que utilizam HTML, CSS e JavaScript. É baseado praticamente em modelos de design para a tipografia e contribui para uma melhor experiência do usuário em um site amigável e, principalmente, responsivo.



Saiba mais

Para saber mais sobre esse assunto, recomendamos a leitura da documentação do *bootstrap* indicada a seguir.

GETBOOTSTRAP. *Introduction*. [s.d.]. Disponível em: <https://bit.ly/3FxyS4J>. Acesso em: 10 maio 2022.

A pseudoclasse `:enabled` representa o contrário lógico de `:disabled` e aplica o estilo fornecido a um elemento que não está desativado. Um elemento é classificado como ativado quando, por exemplo, não há motivo para desativá-lo ou, especificamente, quando não possui um atributo `disabled` aplicado. Deve-se observar que um elemento também pode ser classificado como desativado se a propriedade boolean (*true/false*) desativada do elemento estiver configurada como *true* em seu objeto de documento utilizando JavaScript. Como em outros seletores, as responsabilidades de habilitar e de desabilitar podem ser feitas explicitamente com base no estado do elemento.

No exemplo da figura a seguir, nossa folha de estilos especificará um fundo amarelo com borda azul para todos os elementos `<input>` habilitados que, no caso, serão o campo que recebe o e-mail do usuário e o botão de submissão.



```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       input:enabled
6       {
7         background-color: yellow;
8         border: 2px solid green;
9       }
10    </style>
11  </head>
12  <body>
13    <form action="#" method="post">
14      <label for="txtUserName">User Name</label><br />
15      <input type="text" id="txtUserName" name="txtUserName" disabled /><br /><br />
16      <label for="txtEmail">Email</label><br />
17      <input type="text" id="txtEmail" name="txtEmail" /><br /><br />
18      <input type="submit" value="Submit" />
19    </form>
20  </body>
21 </html>
```

Figura 137 – Exemplo de regra CSS para estilizar elementos ativados



Observação

Em ciências exatas e tecnológicas, *boolean* pode ser definido como um tipo de dado primitivo que possui dois valores (0 – falso e 1 – verdadeiro). A denominação *boolean* é uma honrosa homenagem a George Boole, que, diante de seus estudos, determinou um sistema de lógica algébrica pela primeira vez em meados do século XIX.



Observação

As pseudoclasses `:enabled` e `:disabled` podem ser utilizadas com qualquer elemento HTML, mas são comumente utilizadas em elementos de formulários.

Pseudoclasse `:checked`

A pseudoclasse `:checked` é utilizada em seletores que atuam com elementos de entrada de caixa de seleção e *radio buttons*. Sua premissa é simples: se o elemento estiver marcado ou selecionado, essa regra será aplicada; caso contrário, não será.

Como as caixas de seleção e os botões de opção não trabalham com alteração de cor, utilizaremos uma estratégia diferente para o próximo exemplo: iremos alterar o tamanho dos controles. Observe o código da figura a seguir.

```

C:\Users\salat\Documents\Livros\livros\exercicios_DS\exercicio_1.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

exercicio_1.html x
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <style>
5       input[type="checkbox"], input[type="radio"]
6       {
7         width: 20px;
8         height: 20px;
9       }
10      input[type="checkbox"]:checked, input[type="radio"]:checked
11      {
12        width: 25px;
13        height: 25px;
14      }
15    </style>
16  </head>
17  <body>
18    <form action="#" method="post">
19      <label>Check box 1<input type="checkbox" /></label><br />
20      <label>Check box 2<input type="checkbox" /></label><br />
21      <label>Check box 3<input type="checkbox" /></label><br />
22      <label>Check box 4<input type="checkbox" /></label><br />
23      <label>Radio Button 1<input type="radio" name="radios" /></label><br />
24      <label>Radio Button 2<input type="radio" name="radios" /></label><br />
25      <label>Radio Button 3<input type="radio" name="radios" /></label><br />
26      <label>Radio Button 4<input type="radio" name="radios" /></label><br />
27    </form>
28  </body>
29 </html>

```

Figura 138 – Exemplo de estilos CSS que demonstram o uso da pseudoclasse `:checked`

Na marcação HTML, há alguns elementos de caixa de seleção e *radio buttons* dentro do elemento `<form>`. Na folha de estilos, que está em destaque, utilizamos uma combinação de seletores de atributo com a pseudoclasse `:checked`. Note, também, que fizemos agrupamentos de seletores separando-os por vírgulas, pois, nesse caso, mais de um seletor utiliza cada bloco de declaração.

O seletor `input[type="checkbox"]` seleciona elementos `<input>` cujo atributo `type` possui o valor *checkbox*, enquanto o seletor `input[type="radio"]` seleciona elementos `<input>` cujo atributo `type` possui o valor *radio*. Esses dois seletores de atributo compartilham o mesmo bloco de declaração, que especifica largura e altura de 20 px para os elementos selecionados.

Por sua vez, o seletor `input[type="checkbox"]:checked` seleciona elementos `<input>` cujo atributo `type` possui o valor *checkbox* e que estejam necessariamente marcados pelo usuário durante a execução. O seletor `input[type="radio"]:checked` faz o mesmo, mas para elementos de valor *radio*. Esses dois seletores de atributo com pseudoclasse compartilham o mesmo bloco de declaração, que especifica largura e altura de 25 px para os elementos selecionados.

Quando executado no navegador, o código tem o resultado mostrado na figura a seguir. Conforme mudamos a seleção dos elementos durante a execução, o tamanho dos elementos deve variar.

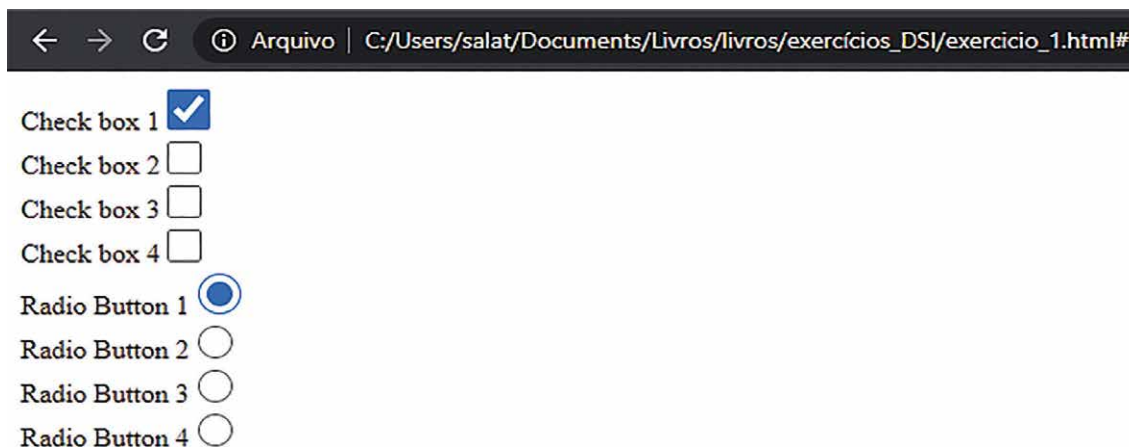


Figura 139 – Resultados do exemplo de estilos CSS que mostram o uso da pseudoclasse `:checked`

A utilização de seletores como esses é o método utilizado, por exemplo, pelos kits de ferramentas da interface do usuário, como o *materialize* e o *bootstrap*, para produzir botões de alternância e de rádio personalizado e verificar grupos desenvolvidos a partir de listas de ícones. Caso desejássemos criar caixas de seleção personalizadas, bastaria combinar esse padrão com pseudoelementos, como antes e depois, para assim adicionar/remover conteúdo dinamicamente na sua marcação.



Saiba mais

O *materialize* é um *framework front-end* responsivo baseado totalmente em *material design*. No entanto, o que diferencia o *materialize* de outros *frameworks front-end*, como o *bootstrap*, é que o *materialize* tem como base o *material design* do Google, enquanto o *bootstrap* tem como base de construção a primeira linguagem de design para dispositivos móveis e na linguagem de design plano. Para saber mais sobre esse assunto, recomendamos a leitura indicada a seguir.

Disponível em: <https://materializecss.com/>. Acesso em: 10 maio 2022.

Além dos tipos de seletores abordados neste livro-texto, existem os seletores de pseudo-elementos, que permitem a estilização de uma parte específica do elemento selecionado. Você pode ler sobre esse assunto no conteúdo disponível no link a seguir.

W3SCHOOLS. *CSS Pseudo-elements*. 2022. Disponível em:

<https://bit.ly/3L5C3Sm>. Acesso em: 10 maio 2022.

6 ESTUDO DE CORES E FONTES

Entre as propriedades CSS mais importantes, estão as que mexem com cores e fontes das nossas páginas HTML. Este tópico será dedicado a algumas informações específicas a respeito desses temas.

6.1 Estudo de cores

Vamos aprender a representar valores de cores nos nossos códigos CSS utilizando diferentes sistemas:

- RGB.
- Hexadecimal.
- Por nome.

6.1.1 Representação em sistema RGB

No contexto computacional, um valor de cor é composto por 256 (0 a 255) níveis de vermelho, 256 (0 a 255) níveis de verde e 256 (0 a 255) níveis de azul, com 0 sendo nenhuma cor e 255 sendo o valor máximo da cor. Dessa forma, os códigos de cores computacionais são uma mistura de cores primárias RGB – sendo o vermelho representado por R (do inglês, *red*), o verde representado por

G (*green*) e o azul representado por B (*blue*). O principal propósito do sistema RGB é possibilitar a reprodução de cores em dispositivos eletrônicos.

No CSS3, uma cor pode ser expressa no sistema RGB utilizando a seguinte sintaxe:

```
rgb(red, green, blue)
```

Cada palavra que representa uma cor deve ser substituída por um número de 0 a 255. Por exemplo, se quiséssemos representar um cinza médio, deveríamos misturar as três cores primárias, utilizando um valor médio entre o mínimo e o máximo para cada uma delas. Poderíamos, portanto, utilizar 127 para os valores de vermelho, verde e azul. Dessa forma, teríamos a representação `rgb(127, 127, 127)`.

Também é possível utilizar taxas percentuais, em vez de valores absolutos, para a representação dos níveis de cor. No caso, o valor 0 corresponde a 0% e o valor 255 corresponde a 100%. A mesma cor cinza médio, cujo código RGB foi apresentado por `rgb(127, 127, 127)`, também pode ser escrita como `rgb(50%, 50%, 50%)`.

6.1.2 Representação em sistema hexadecimal

No sistema numérico decimal, que utilizamos no nosso cotidiano, existem 10 símbolos para a representação de algarismos, contabilizados de 0 a 9. Já no sistema numérico hexadecimal, existem 16 algarismos distintos, contabilizados de 0 a F. Podemos representar cada um dos algarismos hexadecimais em 4 bits (dígitos binários). A representação desses 16 algarismos nos sistemas hexadecimal, decimal e binário é apresentada na tabela a seguir.

Tabela 1

Hexadecimal	Decimal	Binário (4 bits)
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111

Nosso objetivo aqui não é transformar isso em uma lição de matemática hexadecimal, pois esse assunto é abordado em outras disciplinas do curso. Fizemos essa introdução apenas para contextualizar o que será explicado a seguir.

Se dobrarmos a quantidade de bits considerados na tabela, os 4 bits se tornarão 8 bits. Com 8 bits, o número máximo que poderíamos representar no sistema decimal é 255, que representa FF no sistema hexadecimal.

Para representarmos uma cor do sistema RGB utilizando o sistema hexadecimal, utilizamos o símbolo # seguido de seis dígitos hexadecimais, sendo cada dupla de dígitos dedicada a uma das cores primárias (vermelho, verde e azul). A sintaxe do código hexadecimal é representada, portanto, desta forma: #RRGGBB.

No tópico anterior, escrevemos, no formato RGB, uma cor cinza médio, utilizando 127 para os valores de vermelho, verde e azul – 127 (em sistema decimal) representa 7F em hexadecimal. O código da mesma cor cinza médio em hexadecimal, portanto, é #7F7F7F.

Se nos depararmos com o código de cor #FF0000, deduzimos facilmente que se trata de vermelho, já que os dígitos RR se encontram em seu valor máximo (FF), enquanto os dígitos GG e BB estão zerados.

6.1.3 Representação por nomes das cores

O CSS2 introduziu nomes próprios para algumas cores, mas elas não foram amplamente divulgadas. Isso, no entanto, foi bastante expandido com o CSS3. Hoje, o CSS3 reconhece 140 nomes de cores, que podem ser visualizadas na figura a seguir.

aliceblue	antiquewhite	aqua	aquamarine	azure
beige	bisque	black	blanchedalmond	blue
blueviolet	brown	burlywood	cadetblue	chartreuse
chocolate	coral	cornflowerblue	cornsilk	crimson
cyan	darkblue	darkcyan	darkgoldenrod	darkgray
darkgreen	darkkhaki	darkmagenta	darkolivegreen	darkorange
darkorchid	darkred	darksalmon	darkseagreen	darkslateblue
darkslategray	darkturquoise	darkviolet	deeppink	deepskyblue
dimgray	dodgerblue	firebrick	floralwhite	forestgreen
fuchsia	gainsboro	ghostwhite	gold	goldenrod
gray	green	greenyellow	honeydew	hotpink
indianred	indigo	ivory	khaki	lavender
lavenderblush	lawngreen	lemonchiffon	lightblue	lightcoral
lightcyan	lightgoldenrodyellow	lightgray	lightgreen	lightpink
lightsalmon	lightseagreen	lightskyblue	lightslategray	lightsteelblue
lightyellow	lime	limegreen	linen	magenta
maroon	mediumaquamarine	mediumblue	mediumorchid	mediumpurple
mediumseagreen	mediumslateblue	mediumspringgreen	mediumturquoise	mediumvioletred
midnightblue	mintcream	mistyrose	moccasin	navajowhite
navy	oldlace	olive	olivedrab	orange
orangered	orchid	palegoldenrod	palegreen	paleturquoise
palevioletred	papayawhip	peachpuff	peru	pink
plum	powderblue	purple	red	rosybrown
royalblue	saddlebrown	salmon	sandybrown	seagreen
seashell	sienna	silver	skyblue	slateblue
slategray	snow	springgreen	steelblue	tan
teal	thistle	tomato	turquoise	violet
wheat	white	whitesmoke	yellow	yellowgreen

Figura 140 – Os 140 nomes de cores CSS3

Em qualquer lugar que precisarmos especificar uma cor em regras de CSS, poderemos utilizar esses nomes de cores, como fizemos nos exemplos desta unidade.

É claro que cada uma dessas cores nomeadas pode, também, ser representada nos códigos CSS por meio do sistema RGB ou hexadecimal. Vamos utilizar como exemplo a cor de nome red (vermelho). Podemos representá-la de diversas formas, conforme indicado a seguir.

- Pelo nome: red.
- Pelo sistema RGB: rgb(255, 0, 0).
- Pelo sistema RGB (percentual): rgb(100%, 0%, 0%).
- Pelo sistema hexadecimal: #FF0000.

Todos esses quatro valores, quando utilizados em qualquer lugar no código CSS em que uma cor precise ser especificada, resultarão na exibição da mesma cor.



Saiba mais

Se em alguns momentos do desenvolvimento você não se lembrar dessas cores, acesse dois links interessantes para auxiliá-lo na pesquisa.

Disponível em: <https://bit.ly/3PeaN7J>. Acesso em: 10 maio 2022.

COYIER, C. Named colors and hex equivalents. *CSS-Tricks*, jan. 2012. Disponível em: <https://bit.ly/3L1KDSe>. Acesso em: 10 maio 2022.

Vale notar que o site CSS-Tricks não lista apenas as cores, mas também os valores de #rrggbb para cada uma delas.

6.2 Estudo de fontes

Durante quase toda a vida útil da especificação HTML/CSS, muitos desenvolvedores e designers da web ficaram restritos ao mesmo conjunto principal de fontes para sua tipografia de atuação. Caso quiséssemos utilizar determinada fonte, frequentemente precisávamos preparar o texto em um pacote artístico ou processador de texto que tivesse acesso à fonte desejada e, na sequência, salvar a imagem como um arquivo de imagem compatível com o browser, ou até mesmo como uma captura de tela do seu processador de texto para utilizar no documento HTML.

Erros de ortografia e alterações simples de texto levavam um tempo longo para serem corrigidos, e isso era uma tarefa árdua para qualquer outra coisa além dos banners ou manchetes simples.

Embora houvesse uma grande quantidade de frustrações em torno do uso de fontes, havia também boas e sólidas razões para restringir as coisas. Essas razões basicamente se resumiam a diferenças de plataforma.

Os browsers antigos não eram tão capazes quanto os browsers de hoje, e muitos sistemas operacionais tinham apenas uma quantidade limitada de fontes instaladas. Por isso, os comitês de HTML formularam uma lista de fontes conhecidas que estavam praticamente garantidas em qualquer plataforma com qualquer tipo de fonte e browser gráfico.

Não trataremos apenas de Windows e Mac, pois, no início dos anos da década de 1990, havia mais variações de Unix e Linux e de suas distribuições em seus ambientes desktop específicos. Então, nessa mesma época, iniciou-se uma padronização utilizando itens como X11 e gerenciadores de janelas, como Gnome e KDE, que auxiliaram nesse trabalho.



Observação

Gnome é um projeto de software livre que atende, como áreas de atuação, o ambiente de trabalho, de mesmo nome (para os usuários), e a sua plataforma de desenvolvimento (para os desenvolvedores). O projeto dá ênfase à usabilidade, à acessibilidade e à internacionalização.

KDE é uma comunidade internacional de software livre que desenvolve um conjunto de aplicativos multiplataforma desenhados e customizados para o funcionamento em conjunto com sistemas, tais como Linux, Solaris, Microsoft Windows e Apple Mac OS X.

Entre as fontes básicas restritas que quase sempre garantiam o trabalho, tínhamos:

- Sans-serif
- Serif
- Fantasy
- Cursive
- Monospace

Se quiséssemos especificar nossa fonte utilizando uma dessas opções, primeiramente obteríamos a versão dessa plataforma e, embora nem sempre sejam exatamente idênticos, a aparência, o estilo e a sensação estariam próximos o suficiente para que o layout do nosso design não fosse muito afetado nem prejudicado no resultado de sua apresentação.

Dependendo da plataforma de uso, também seria possível disponibilizar substituições a serem utilizadas. Assim, geralmente, em sites desenvolvidos em máquinas Windows, veríamos fontes descritas como Times, Times New Roman e Serif.

Por exemplo, se especificássemos Impact Bold, Sans-serif, qualquer plataforma que tivesse Impact Bold instalada seria exibida como pretendido; ou seja, se a fonte não estivesse disponível, a Sans-serif seria utilizada como padrão.

Não existia uma fonte original, e muitos designs simplesmente desmoronavam pelo fato de não renderizarem corretamente. Porém, com o CSS3, houve uma implementação de correção, que fornece a capacidade de empacotar fontes em diferentes formatos com seu aplicativo da web e de fazer o download do servidor e utilizá-las para detectar se a plataforma de destino não as tem disponíveis (ou mesmo no caso de não estarem aderentes).

Isso não significa que não se deva utilizar a lista de fontes seguras da web, pois basta uma perda de conectividade para retornarmos à estaca zero.

6.2.1 Gerando uma fonte adequada

Talvez a maneira mais eficiente de usar uma fonte para o design seja utilizar o site Font Squirrel, que pode ser empregado para converter uma fonte do seu sistema em um kit instantaneamente utilizável de regras e folhas de estilo CSS3.



Saiba mais

Você pode baixar suas fontes no Font Squirrel, indicado a seguir.

FONT SQUIRREL. *Webfont Generator*. 2022. Disponível em: <https://bit.ly/3M5sZhB>. Acesso em: 10 maio 2022.

Utilizar o site indicado anteriormente é muito simples: você precisa clicar, adicionar fontes e selecionar os arquivos que deseja adicionar. Depois de adicionar as fontes que quer converter, clique na caixa de contrato para continuar. Assim, o Font Squirrel irá converter as fontes, conforme apresentado a seguir.

The screenshot shows the Font Squirrel Webfont Generator website. The browser address bar displays 'fontquirrel.com/tools/webfont-generator'. The website has a dark purple header with the 'FONT SQUIRREL' logo and a navigation menu with links: 'Hot', 'Recent', 'Top Font Deals', 'Swag', 'Generator' (highlighted), 'Font Identifier', 'Font Talk', and 'Blog'. A pink banner for 'Creative Cloud for teams' is visible. The main content area is titled 'Webfont Generator' and includes a 'Usage' note. A red box highlights the 'UPLOAD FONTS' button. Below it, a message states 'You currently have no fonts uploaded.' There are three radio button options: 'BASIC' (selected), 'OPTIMAL', and 'EXPERT...'. An 'Agreement' section contains a checked checkbox and the text: 'Yes, the fonts I'm uploading are legally eligible for web embedding. Font Squirrel offers this service in good faith. Please honor the EULAs of your fonts.'

Figura 141 – Representação do upload da fonte

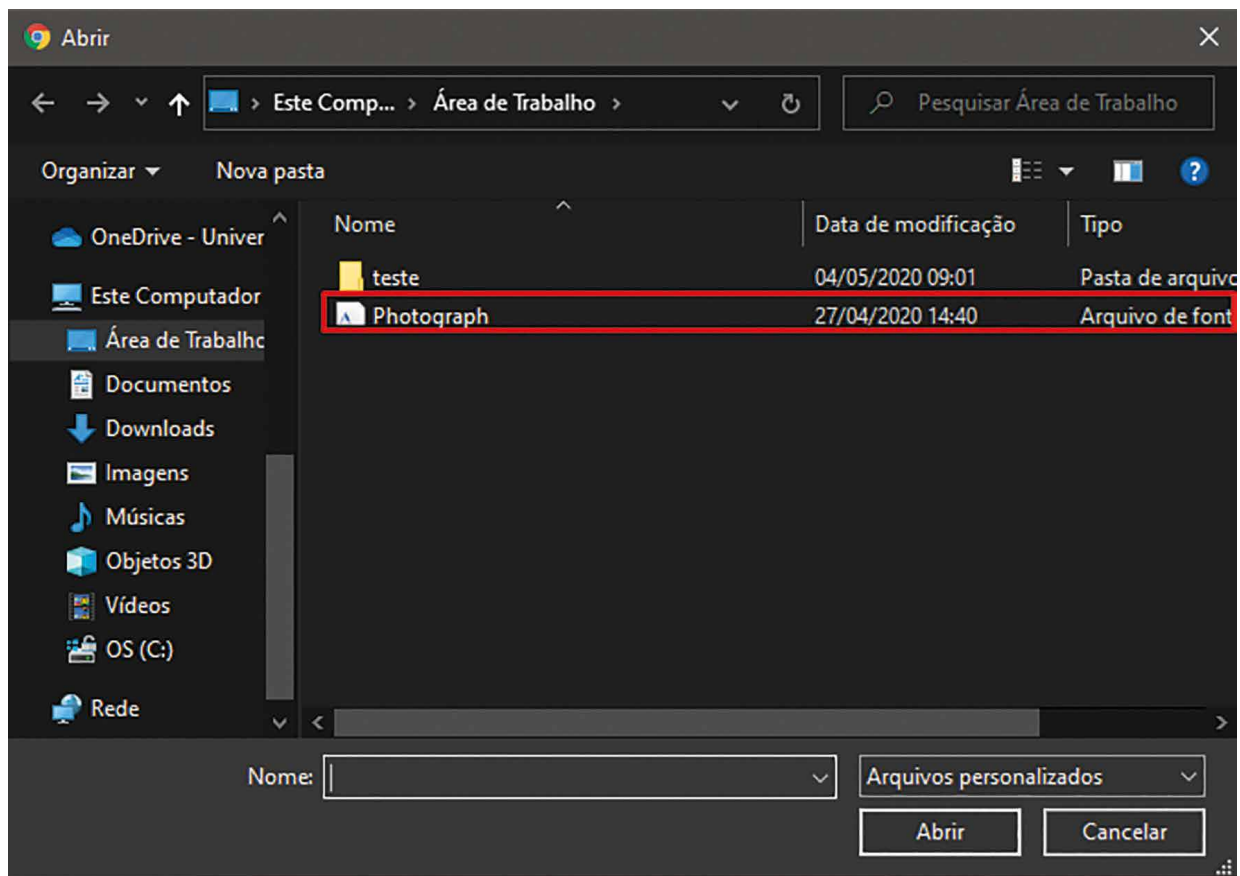


Figura 142 – Localizando a fonte



Saiba mais

Para ver na prática o exemplo anterior, em que a fonte Photograph foi usada, indicamos que acesse o site a seguir.

Disponível em: <https://www.dafont.com/pt/>. Acesso em: 10 maio 2022.

Quando finalizar, um link de download será apresentado para um arquivo zip, que contém a fonte em diversos formatos compatíveis com a web, além de diversos arquivos CSS e exemplos de marcação HTML mostrando como utilizá-lo, conforme apresentado a seguir.



Figura 143 – Representação do download do kit

Outra possibilidade é a utilização de fontes do diretório de fontes abertas do Google Fonts, que faz questão de utilizar essa ferramenta para possibilitar que todas as fontes listadas no site sejam utilizáveis nos designs de páginas web.



Saiba mais

Com o Google Fonts, podemos pesquisar, filtrar e comparar qualquer uma das fontes da lista. Depois de escolher a fonte que desejamos utilizar, podemos adicioná-la a uma coleção ou ver exemplos rápidos de sua utilização.

Sugerimos que você visite o site a seguir.

Disponível em: <https://fonts.google.com/>. Acesso em: 10 maio 2022.

A página de utilização rápida é particularmente útil porque lista tudo o que precisamos saber para adicionar de forma prática e ágil essa fonte à nossa página web.

Seguindo as instruções na página de utilização rápida e copiando e colando o código de amostra fornecido, podemos utilizar a fonte diretamente das redes globais de entrega de conteúdo do Google, para que seja carregada de modo rápido, sem que você precise se preocupar com o download e adicioná-la ao seu próprio servidor web.

Se optarmos por adicioná-la a uma coleção, podemos continuar adicionando outras fontes, e tudo aparecerá no nosso painel Coleção, na parte inferior da página, conforme ilustrado a seguir.

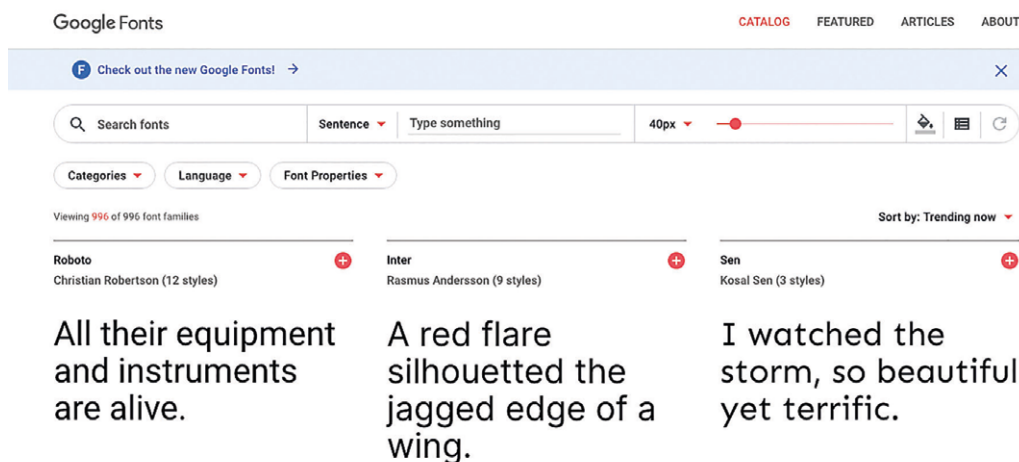


Figura 144 – Representação do Google Fonts



Resumo

Nesta unidade, vimos como podemos combinar e integrar técnicas de desenvolvimento web, como o CSS, que, de fato, enriquecem todo o desenvolvimento de aplicação web, principalmente quando se trata de sua recursividade (bootstrap). Com isso, disponibilizamos a visualização de páginas tanto em desktop quanto em smartphones e tablets, sem que haja perda de sua qualidade.

Conceitos de CSS aprimoraram o desenvolvimento de aplicação web e do trabalho de aplicação de suas propriedades. Os exemplos apresentados ao longo da unidade têm como premissa a utilização de um código semelhante e procuram mostrar como desenvolver sobre esse código.

Ressaltamos que, assim como as novas especificações do HTML5, o CSS3 (ou CSS versão 3, para sermos mais precisos) pode ser definido como o conjunto mais atualizado de especificações projetadas para moldar, modelar e definir quais recursos a versão mais recente do CSS apresenta atualmente.

Por fim, no estudo de cores, observamos que, no esquema numérico hexadecimal, os dígitos são definidos e contabilizados de 0 a F, em vez de 0 a 9, como ocorre no sistema numérico decimal. Logo, quando observamos atentamente, notamos que podemos ajustar todos esses valores em 4 bits. Como esse número máximo pode ser representado em decimal como o 15, confirmamos que 0 a F realmente significa 0 a 15.



Exercícios

Questão 1. (FGV 2018) Analise o código HTML a seguir.

```
<!DOCTYPE html>
<html>
<head>
<style>
</style>
</head>
<body>
<h1>Título</h1>
<p>Primeiro parágrafo</p>
</body>
</html>
```

Para que o título seja exibido na cor azul, o código CSS inserido entre a quarta e a quinta linha deve ser:

- A) font = "blue" for h1
- B) h1:color="blue"
- C) <h1><color>blue</color></h1>
- D) {h1}<color:blue>{/h1}
- E) h1 {color:blue}

Resposta correta: alternativa E.

Análise da questão

A questão pede a inserção de um código CSS no formato interno, ou seja, entre as tags <style> e </style> do próprio arquivo HTML. A sintaxe básica de uma regra de estilo é:

```
seletor {
propriedade1: valor;
propriedade2: valor;
...
}
```

O seletor identifica os elementos que serão afetados pela regra de estilo. Como a questão pede uma regra para um título, o seletor, no caso, seria h1. A propriedade que ajusta a cor do texto é a color, que deve assumir o valor blue. Aplicando essa regra no formato da sintaxe básica, teríamos:

```
h1 {  
  color: blue;  
}
```

No entanto, em se tratando de HTML e CSS, não faz diferença escrever um trecho de código em apenas uma linha ou em várias linhas, pois ele será interpretado pelo navegador do mesmo jeito. Além disso, o ponto e vírgula é facultativo quando a regra CSS for constituída por apenas uma declaração. Dessa forma, podemos realizar a declaração no formato a seguir:

```
h1 {color:blue}
```

Questão 2. (FGV 2019, adaptada) Analise o trecho de HTML/CSS exibido a seguir.

```
<head>  
<style>  
div.ex1 { width:800px; }  
div.ex2 { max-width:800px; }  
</style>  
</head>  
<body style="font-size: 24px">  
<div class="ex1"> Este texto ocupa  
aproximadamente uns 15 centímetros da  
tela.</div>  
<div class="ex2">Este texto ocupa  
aproximadamente uns 15 centímetros da  
tela.</div>  
<div>Este texto ocupa aproximadamente uns 15  
centímetros da tela.</div>  
</body>
```

Num display com largura de 30 cm, com a janela do browser no tamanho normal, numa página contendo apenas esses elementos, cada div é exibido em uma única linha.

Nesse cenário, considere as hipóteses sobre o comportamento dos elementos quando a largura da janela do browser é reduzida para 6 cm.

- I – A largura do primeiro div é reduzida, e o texto é rearranjado de acordo.
- II – A largura do segundo div mantém-se, e parte do texto torna-se não visível.
- III – A largura do terceiro div é reduzida, e o texto é rearranjado de acordo.

Sobre essas hipóteses, conclui-se que:

- A) Somente I e II são verdadeiras.
- B) Somente I e III são verdadeiras.
- C) Somente II e III são verdadeiras.
- D) Somente III é verdadeira.
- E) I, II e III são verdadeiras.

Resposta correta: alternativa D.

Análise das afirmativas

I – Afirmativa incorreta.

Justificativa: o primeiro elemento <div> do código será afetado pela seguinte declaração de estilo:

```
div.ex1 { width:800px; }
```

O seletor dessa declaração indica que os elementos <div> que têm o atributo class="ex1" serão selecionados.

Dessa forma, quando exibido no navegador, esse elemento manterá uma largura fixa de 800 px, independentemente do tamanho da janela. Quando a janela é ajustada para uma largura menor do que essa, parte do texto exibido torna-se não visível ao usuário.

II – Afirmativa incorreta.

Justificativa: o segundo elemento <div> do código será afetado pela seguinte declaração de estilo:

```
div.ex2 { max-width:800px; }
```

O seletor dessa declaração indica que os elementos <div> que têm o atributo class="ex2" serão selecionados.

Dessa forma, quando exibido no navegador, esse elemento exibirá uma largura máxima de 800 px, mas será reduzida quando a janela for ajustada para uma largura menor.

III – Afirmativa correta.

Justificativa: o terceiro elemento <div> do código não será afetado por qualquer declaração de estilo presente entre as tags <style> e </style>, já que não foi declarado seu atributo de classe. Dessa forma, o elemento assume o estilo padrão, que ajusta o conteúdo do elemento à largura da janela do navegador.