1 Introdução

As páginas que visitamos são escritas utilizando-se HTML, que é uma linguagem apropriada para a especificação de sua **estrutura**. Por exemplo, usamos HTML para especificar que uma página possui um **cabeçalho principal**, uma **seção principal**, um **rodapé**, duas **seções laterais** etc. A linguagem CSS, por sua vez, é utilizada para especificar o **estilo** das páginas. **Cores**, **espaçamento**, **posicionamento**, **bordas** e muitas outras características de uma página são especificadas utilizando-se CSS.

Antes de prosseguirmos, vejamos algumas informações importantes.

I. **CSS** é sigla para **Cascading Style Sheets** (algo como folhas de estilo em cascata). Entenderemos esse nome em breve.

II. A primeira versão de CSS, chamada CSS 1, foi lançada em 1996.

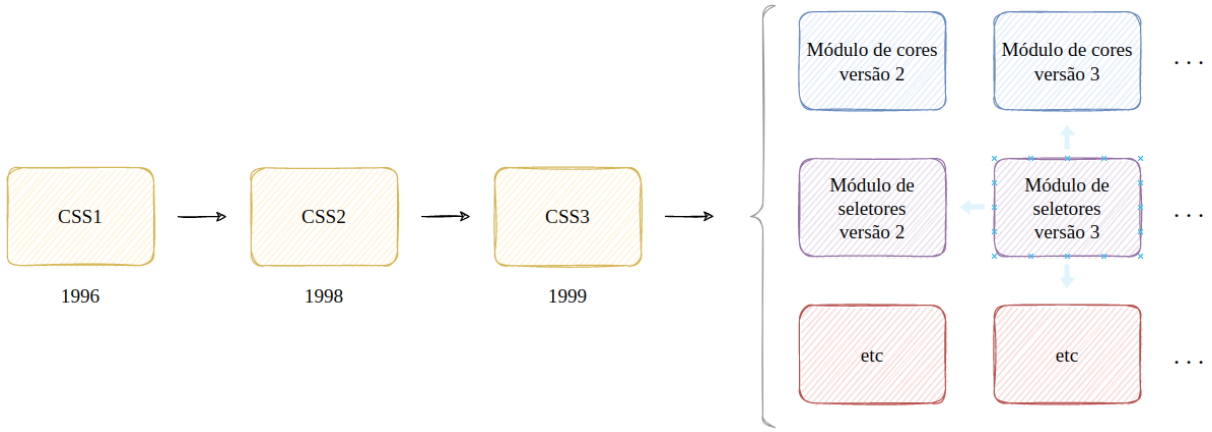
III. Em 1998, foi lançada a segunda versão de CSS, chamada CSS 2.

IV. CSS3 é a última versão de CSS e foi lançada em 1999.

V. CSS3 se encontra em contínuo desenvolvimento desde então.

VI. CSS3 é dividida em **módulos**. Módulos para cor, para sombreamento etc. Cada módulo tem seu próprio esquema de versionamento. A ideia é prosseguir adicionando novos módulos e aprimorar cada um deles, dando origem a novas versões de módulos. Por isso, **não há CSS4** ou algo parecido com isso. Veja a Figura 1.1.

Figura 1.1



VII. Visite o Link 1.1 para ver a lista de especificações de CSS.

Link 1.1
https://www.w3.org/Style/CSS/specs.en.html

Por curiosidade, veja o que a página fala sobre a primeira versão de CSS:

> *Level 1 contains just the most basic properties of CSS, such as 'margin', 'padding', 'background', 'color' and 'font', with restrictions on the allowed values. It was the first level of CSS to be completed (in 1996) and matched the capabilities of implementations of the time. It is currently only of historical interest; all implementations should be able to support level 2 and probably large parts of level 3, too.*

VIII. É importante observar que, conforme a especificação de CSS evolui, os navegadores as implementam pouco a pouco. Visite o Link 1.2 para verificar como se encontram as principais implementações no momento.

Link 1.2
https://en.wikipedia.org/wiki/Comparison_of_browser_engines_(CSS_support)

2 Desenvolvimento

Passamos a estudar os recursos de CSS de maneira prática.

Nota. Caso esteja utilizando o VS Code, a extensão **Live Server** de **Ritwick Dey** pode ser interessante. Ela permite a visualização das páginas desenvolvidas. Para utilizá-la, colocamos um servidor local em funcionamento que se encarrega de atualizar a página automaticamente conforme salvamos os arquivos que estamos editando.

Nota. A escrita de CSS requer apenas um editor de texto. Apesar disso, há diversas ferramentas que podem ser de interesse. Veja o Link 2.1.

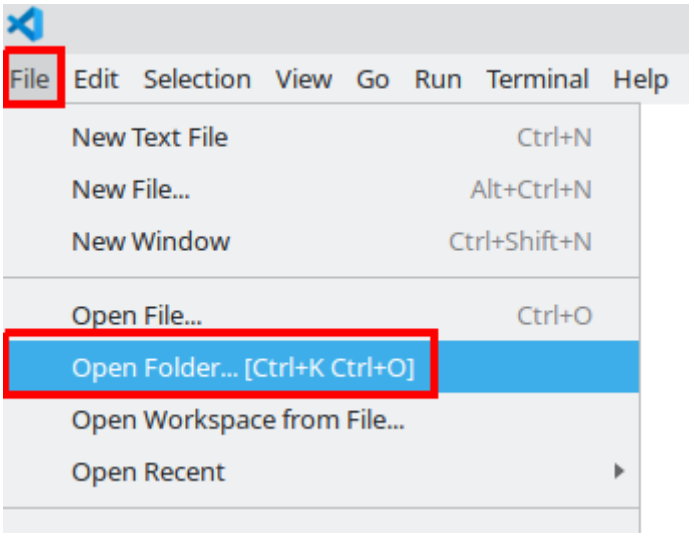
Link 2.1
https://www.w3.org/Style/CSS/software#editors

(Criando uma pasta e um arquivo) Crie uma pasta para os próximos experimentos. Caso esteja utilizando o sistema operacional Windows, uma sugestão é utilizar

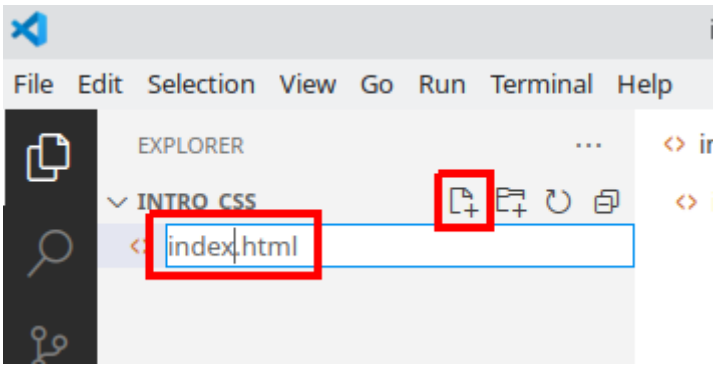
C:\Users\seuUsuario\Documents\intro_css

Tome sempre cuidado para não utilizar pastas que estejam sob o efeito do OneDrive, Dropbox ou outros.

A seguir, vincule o VS Code à pasta criada clicando em File >> Open Folder:



A seguir, crie um arquivo chamado **index.html**:



Veja o conteúdo inicial do arquivo a seguir. Ele ainda não possui código CSS algum. Vamos usá-lo como um primeiro exemplo para aplicar estilos CSS.

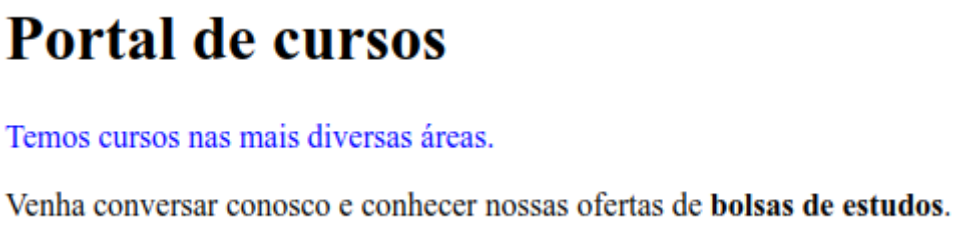
```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
</head>

<body>
 <h1>Portal de cursos</h1>

 <p>Temos cursos nas mais diversas áreas.</p>

 <p>
   Venha conversar conosco e conhecer nossas ofertas
de <strong>bolsas de estudos</strong>.
 </p>
</body>

</html>
```

(CSS inline) A seguir, vamos alterar a cor de fonte do primeiro parágrafo. Para isso, usamos um atributo que o elemento possui. Neste caso ele se chama **style**. Essa forma de escrever, diretamente no elemento, se chama inline. Logo veremos outras alternativas. Ainda neste exemplo, dizemos que

- **color** é uma **propriedade** CSS
- **blue** é um **valor associado à propriedade** color
- **color: blue;** escritos desta forma, caracterizam uma **regra CSS**

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
 <meta charset="utf-8">
 <title>Cursos</title>
</head>

<body>
 <h1>Portal de cursos</h1>

 <p style="color: blue">Temos cursos nas mais diversas áreas.</p>

 <p>
   Venha conversar conosco e conhecer nossas ofertas de <strong>bolsas de estudos</strong>.
 </p>
</body>

</html>
```

Abra o arquivo index.html utilizando seu navegador preferido para ver o resultado, que deve parecido com o seguinte:



Altere também a cor de fonte dos demais elementos:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
</head>

<body>
 <h1 style="color: green">Portal de cursos</h1>

 <p style="color: blue">Temos cursos nas mais diversas
áreas.</p>

 <p style="color: blue">
   Venha conversar conosco e conhecer nossas ofertas
de <strong>bolsas de estudos</strong>.
 </p>
</body>

</html>
```

(Problemas com Css inline: usando o elemento style e seletores) Note que o css inline apresenta alguns problemas:

- não é possível reutilizar definições. Para deixar a cor de fonte dos dois parágrafos em azul, precisamos escrever o código duas vezes.
- As regras CSS ficam “misturadas” com o código HTML, o que pode tornar mais difícil a manutenção da página

Uma outra possibilidade é fazer as definições desejadas em um **elemento do tipo style**, que deve ser definido como **filho do head da página**. Para isso, passaremos a utilizar os **seletores CSS**. Um seletor permite especificar o elemento ou elementos cujas propriedades desejamos alterar. Veja como fica a seguir. Não se esqueça de remover o CSS inline.

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
 <style>
    /* selecionamos os elementos h1 aqui */
    h1 {
      color: green;
    }
    /* aqui selecionamos os elementos p */
    p{
      color: blue;
    }
 </style>
</head>

<body>
  <h1>Portal de cursos</h1>

 <p>Temos cursos nas mais diversas áreas.</p>

 <p>
   Venha conversar conosco e conhecer nossas ofertas
de <strong>bolsas de estudos</strong>.
 </p>
</body>

</html>
```

(Herança de regras) Perceba, também, que **o texto do elemento strong ficou azul, embora não tenhamos especificado isso**. Ocorre que, **algumas propriedades são herdadas pelos elementos filhos**. É o caso, por exemplo, da propriedade color. Caso queiramos, podemos definir outra cor para o strong:

```
…
<title>Cursos</title>
 <style>
   /* selecionamos os elementos h1 aqui */
   h1 {
     color: green;
   }
   /* aqui selecionamos os elementos p */
   p{
     color: blue;
   }

   strong {
     color: red;
   }
 </style>
…
```

(Separando a definição de CSS em um arquivo) Os estilos estão definidos no mesmo arquivo em que o conteúdo HTML está definido. Em geral, isso não é uma boa prática pois compromete o nível de **reusabilidade e manutenabilidade** das definições. Para resolver isso, podemos criar um novo arquivo textual em que as definições serão feitas e importá-lo no arquivo HTML. Assim, ele pode ser importado por diferentes arquivos se necessário.

- Comece criando **uma pasta chamada css no mesmo diretório** em que está o arquivo index.html. A criação da pasta é opcional e o nome é qualquer. É somente uma convenção bastante utilizada no mercado.

- A seguir, **dentro da pasta recém criada, crie um arquivo chamado styles.css**. O nome também é só uma convenção, é possível usar qualquer nome.

- A seguir, **recorte toda a definição feita anteriormente no elemento style e cole no arquivo recém criado, sem incluir o elemento style.**

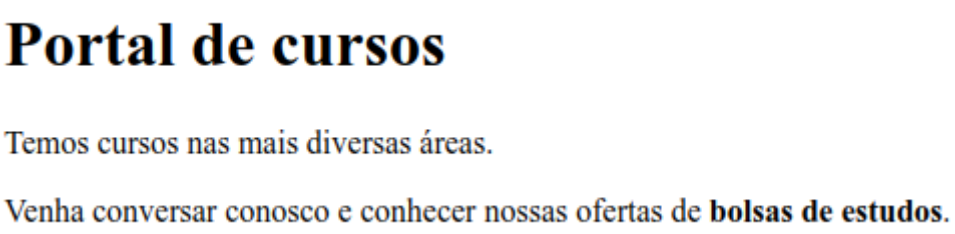
Veja o arquivo **styles.css**:

```
/* arquivo styles.css */
/* selecionamos os elementos h1 aqui */
h1 {
 color: green;
}
/* aqui selecionamos os elementos p */
p{
 color: blue;
}

strong {
 color: red;
}
```

E agora o arquivo **index.html**:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
 <!-- o elemento style foi removido por completo -->
</head>
…
```

Observe que, neste momento, as regras CSS não têm mais efeito sobre os elementos HTML:



Resta saber como acessar o conteúdo definido no arquivo css, a partir do arquivo html. Isso pode ser feito por meio de um elemento chamado **link**, definido como **filho de head**. Em seu atributo **href**, especificamos o endereço do arquivo a ser importado, considerando a pasta em que ele está. Não deixe de apagar o elemento style usado anteriormente.

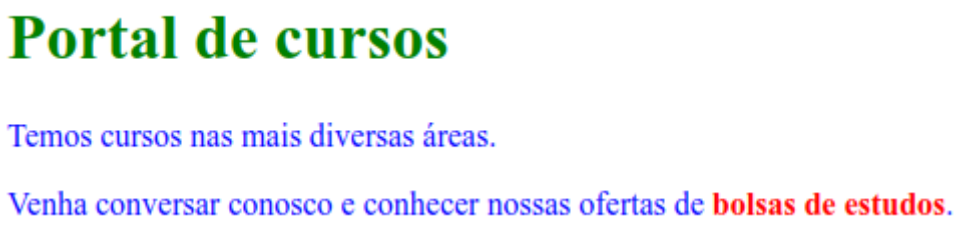
Nota. O Atributo **rel** do elemento **link** vem de "relationship". A ideia é que estamos estabelecendo uma relação entre o arquivo atual e um recurso externo. Embora o elemento link seja muito mais comumente utilizado para importar arquivos CSS, ele também pode ser usado na definição de ícones e outros recursos. Veja mais aqui:

https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link

Veja seu uso no arquivo **index.html**:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
 <link rel="stylesheet" href="css/styles.css">
</head>

<body>
...
```

A página deverá ser exibida já com o efeito das regras CSS novamente:



(Seletores) Estamos utilizando seletores para especificar a quais elementos determinadas regras CSS devem ser aplicadas. Os seletores que utilizamos até então, se baseiam no tipo do elemento. Também podemos utilizar seletores que se baseiam em **classes e em ids**.

(Seletor por id) Para usar um seletor **baseado em id**

- especificamos um identificador para o elemento que desejamos selecionar usando seu atributo id. Depois, no arquivo css, o selecionamos pelo id, precedendo o nome do id pelo símbolo #.

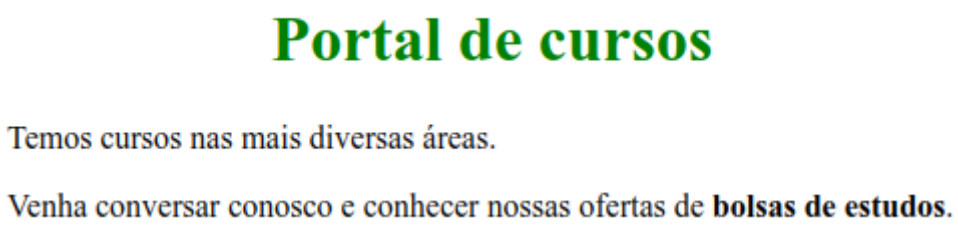
Para este exemplo, **apague todo o conteúdo do arquivo .css**. No arquivo **index.html**, especificamos um id para o elemento h1, por exemplo:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
 <link rel="stylesheet" href="css/styles.css">
</head>

<body>
  <h1 id="titulo">Portal de cursos</h1>
…
```

No arquivo **styles.css**, aplicamos o seletor como descrito. Observe que aproveitamos para centralizar o texto com outra regra CSS.

```
#titulo {
 color: green;
 text-align: center;
}
```

Resultado esperado:



(Seletor por tipo e id) Dá para ser mais específico, dizendo que somente **elementos de um determinado tipo e com um id definido** devem ser selecionados.

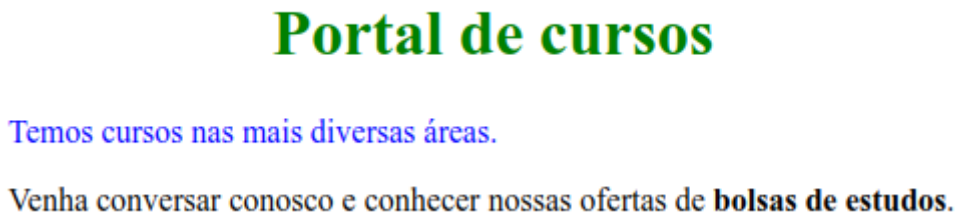
No arquivo **index.html**:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
 <link rel="stylesheet" href="css/styles.css">
</head>

<body>
  <h1 id="titulo">Portal de cursos</h1>

 <!-- somente o primeiro p tem id definido -->
 <p id="descricao">Temos cursos nas mais diversas áreas.</p>
…
```

No arquivo **styles.css**:

```
#titulo {
 color: green;
 text-align: center;
}

/* somente elementos do tipo p que tenham id igual a
descrição */
p#descricao {
 color: blue;
}
```

Resultado esperado:



(Seletor por classe) Elementos HTML têm um atributo chamado class. Para usar um seletor **baseado em class:**

- no arquivo HTML, especificamos um ou mais nomes que ficam associados ao atributo class do elemento.
- no arquivo CSS, para cada nome definido, escrevemos um seletor que consiste de seu nome precedido por um ponto (.)

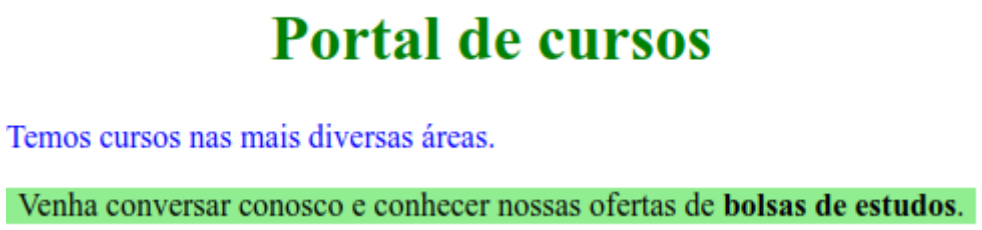
Veja o exemplo a seguir, no arquivo **index.html:**

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Cursos</title>
 <link rel="stylesheet" href="css/styles.css">
</head>

<body>
  <h1 id="titulo">Portal de cursos</h1>

 <!-- somente o primeiro p tem id definido -->
 <p id="descricao">Temos cursos nas mais diversas
áreas.</p>

 <p class="promocao">
   Venha conversar conosco e conhecer nossas ofertas
de <strong>bolsas de estudos</strong>.
 </p>
</body>
</html>
```

No arquivo **styles.css** fica assim:

```
#titulo {
 color: green;
 text-align: center;
}

/* somente elementos do tipo p que tenham id igual a
descrição */
p#descricao {
 color: blue;
}

.promocao {
 text-align: center;
 background-color: lightgreen;
}
```

Resultado esperado:



(Seletor por tipo e classe) De maneira análoga ao que ocorre com o id, também podemos ser mais específicos quando usamos classes, dizendo que somente elementos de um determinado tipo e cuja classe seja alguma que definirmos devem ser selecionados. Veja como fica no arquivo **styles.css**.

```
#titulo {
 color: green;
 text-align: center;
}

/* somente elementos do tipo p que tenham id igual a
descrição */
p#descricao {
 color: blue;
}

p.promocao {
 text-align: center;
 background-color: lightgreen;
}
```

Observe que o resultado visual não mudou neste caso, já que temos apenas um elemento do tipo p ao qual aplicamos a classe promocao e ele já estava sob o efeito dessa classe.

Faça um teste com o seguinte seletor no arquivo **styles.css:**

```
…
h1.promocao {
 color: yellow
}
```

Estamos tentando selecionar todos os elementos do tipo h1 que tenham a classe promocao aplicada a eles. Como não há nenhum, nada mudou graficamente. Pode apagar essa regra depois de realizar o teste.

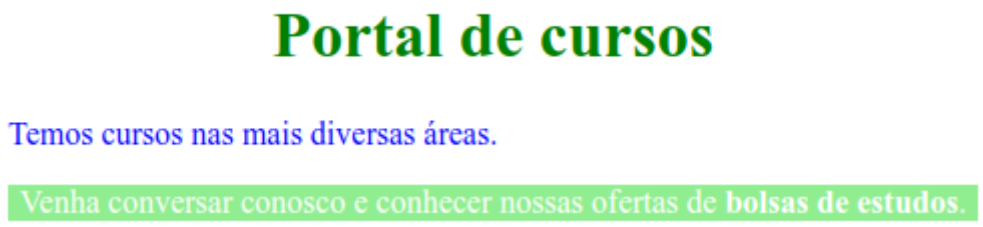
(Múltiplas classes) O atributo class admite um número arbitrário de classes. Para especificá-las, basta separar seus nomes por um espaço em branco. Veja um exemplo no arquivo **styles.css**:

```
#titulo {
 color: green;
 text-align: center;
}

/* somente elementos do tipo p que tenham id igual a
descrição */
p#descricao {
 color: blue;
}

p.promocao {
 text-align: center;
 background-color: lightgreen;
}


.nova {
 color: white;
}
```

No arquivo **index.html** fica assim:

```
…
<body>
  <h1 id="titulo">Portal de cursos</h1>

 <!-- somente o primeiro p tem id definido -->
 <p id="descricao">Temos cursos nas mais diversas
áreas.</p>

 <p class="nova promocao">
   Venha conversar conosco e conhecer nossas ofertas
de <strong>bolsas de estudos</strong>.
 </p>
</body>

…
```

E o resultado esperado se parece com esse:



(Seletores por id, classe e tipo) Existe também a possibilidade de se escrever seletores que envolvem o tipo, a classe e o id simultaneamente. Veja o exemplo a seguir, no arquivo **index.html**:

```
…
<body>
  <h1 id="titulo">Portal de cursos</h1>

 <!-- somente o primeiro p tem id definido -->
 <p id="descricao">Temos cursos nas mais diversas
áreas.</p>

 <p class="nova promocao">
   Venha conversar conosco e conhecer nossas ofertas
de <strong>bolsas de estudos</strong>.
 </p>

 <!-- esse p tem somente classe -->
 <p class="final-pagina">Entre em contato:
11223344</p>
 <!-- esse p tem id e classe. a ordem não importa -->
 <p id="rodape" class="final-pagina">&copy; Todos os
direitos reservados</p>

</body>

…
```

No arquivo **styles.css**:

```
#titulo {
 color: green;
 text-align: center;
}

/* somente elementos do tipo p que tenham id igual a
descrição */
p#descricao {
 color: blue;
}

p.promocao {
 text-align: center;
 background-color: lightgreen;
}


.nova {
 color: white;
}

/* todo p que tenha a classe final-pagina e cujo id
seja rodape */
p.final-pagina#rodape{
 background-color: lightgray;
 text-align: center;
}
```

(Regras repetidas em seletores diferentes: agrupamento de seletores) Suponha que temos duas classes CSS que possuem algumas regras em comum, como a seguir, no arquivo **styles.css**.

```
…
/* todo p que tenha a classe final-pagina e cujo id
seja rodape */
p.final-pagina#rodape{
 background-color: lightgray;
 text-align: center;
}

/* todo curso tem essa cor de fundo */
.curso {
 background-color: aliceblue;
}

/* veja a cor de fundo repetida para que possamos
especificar uma regra não aplicável aos demais cursos
*/
.novo-curso{
 background-color: aliceblue;
 color: red;
}
```

No arquivo **index.html**::

```
…
<body>
  <h1 id="titulo">Portal de cursos</h1>

 <!-- somente o primeiro p tem id definido -->
 <p id="descricao">Temos cursos nas mais diversas
áreas.</p>

 <p class="nova promocao">
   Venha conversar conosco e conhecer nossas ofertas
de <strong>bolsas de estudos</strong>.
 </p>

 <p>Cursos:</p>
 <p class="novo-curso">Ciência da Computação</p>
 <p class="curso">Engenharia de Computação</p>
 <p class="curso">Farmácia</p>

 <!-- esse p tem somente classe -->
 <p class="final-pagina">Entre em contato:
11223344</p>
 <!-- esse p tem id e classe. a ordem não importa -->
 <p id="rodape" class="final-pagina">&copy; Todos os
direitos reservados</p>

</body>

…
```

Perceba que a cor de fundo para ambas as classes é a mesma. Ou seja, elas têm características em comum. E a classe novo-curso possui algumas outras propriedades. Essa definição pode ser reescrita a seguir, agrupando classes de modo que definições comuns a elas sejam escritas uma única vez. O operador próprio para esse agrupamento é a vírgula:

```
…
/* todo p que tenha a classe final-pagina e cujo id
seja rodape */
p.final-pagina#rodape{
 background-color: lightgray;
 text-align: center;
}

/* separamos os seletores usando uma vírgula. Ela faz
um papel de "ou lógico */
.curso , .novo-curso {
 background-color: aliceblue;
}

/* agora não precisamos mais repetir a cor de fundo */
.novo-curso{
 color: red;
}
```

Veja um próximo exemplo de agrupamento de seletores. Temos um seletor por id e dois por classe. No arquivo **styles.css**:

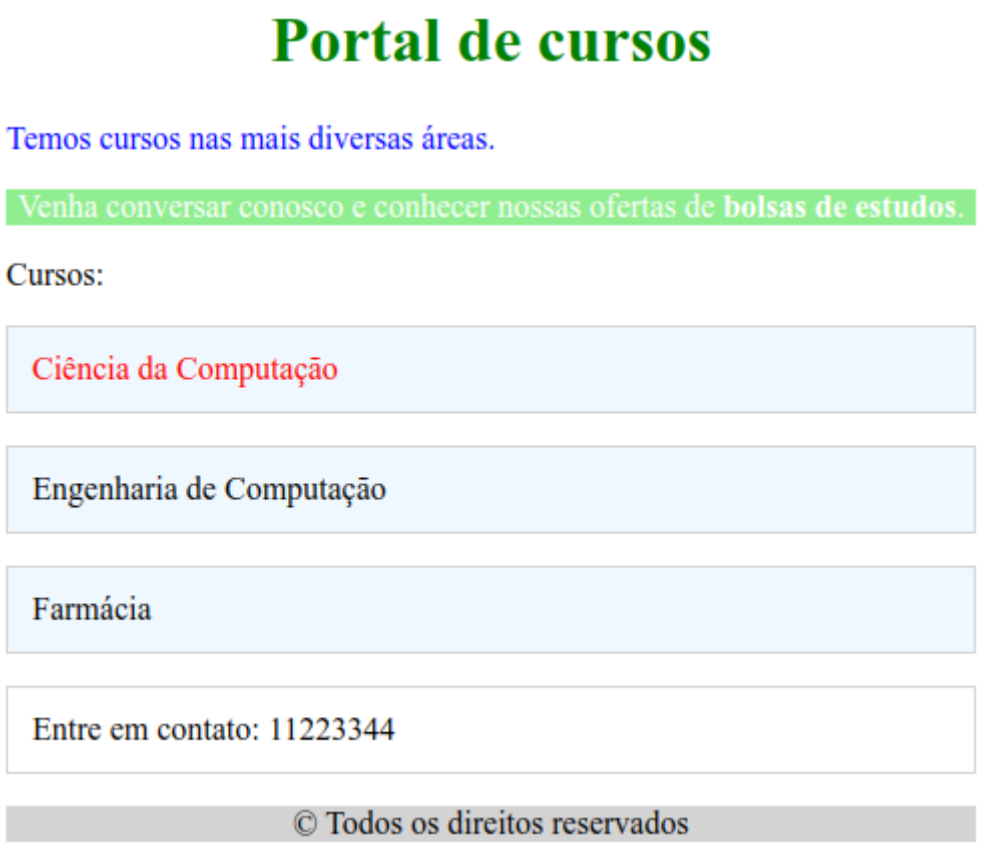
```
…
/* separamos os seletores usando uma vírgula. Ela faz
um papel de "ou lógico */
.curso , .novo-curso {
 background-color: aliceblue;
```

```
}

/* agora não precisamos mais repetir a cor de fundo */
.novo-curso{
 color: red;
}

.curso, .novo-curso, #contato {
 /* padding: espaço entre o conteúdo e a borda.
Veremos mais sobre isso adiante */
 padding: 12px;
 /* estudaremos mais sobre bordas também */
 border: 1px solid lightgray;
}
```

No arquivo **index.html**:

```
…
 <p>Cursos:</p>
 <p class="novo-curso">Ciência da Computação</p>
 <p class="curso">Engenharia de Computação</p>
 <p class="curso">Farmácia</p>

 <!-- esse p tem somente classe -->
 <p id="contato" class="final-pagina">Entre em contato:
11223344</p>
 <!-- esse p tem id e classe. a ordem não importa -->
 <p id="rodape" class="final-pagina">&copy; Todos os
direitos reservados</p>

</body>
…
```

Veja o resultado esperado:



Nota. Por um lado, agrupar seletores evita escrever código repetido, o que simplifica manutenções futuras. Por outro lado, como veremos, em alguns casos estaremos interessados em escrever componentes (elementos HTML decorados com CSS) independentes reutilizáveis, que possam existir em qualquer contexto, de maneira independente dos demais. Neste caso, o agrupamento pode não ser desejado. Há um meio termo, portanto, a ser levado em consideração. Cabe ao programador decidir o que é melhor dentro de cada cenário.

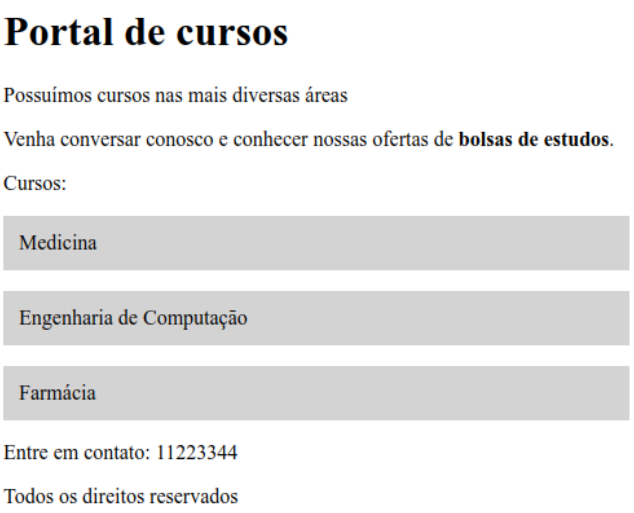
(Seletores descendentes) Para este exemplo, crie um novo arquivo chamado **mais_seletores.css** na raiz do projeto, lado a lado com o arquivo index.html. Veja seu conteúdo inicial. Para simplificar, vamos especificar as regras CSS direto no arquivo HTML, sem usar um arquivo .css, portanto.

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Mais seletores</title>
 <!-- vamos especificar as regras CSS aqui, para simplificar -->
 <style>

 </style>
</head>

<body>
 <main>
   <h1 id="titulo">Portal de cursos</h1>
   <p id="descricao">Possuímos cursos nas mais diversas áreas</p>

   <p class="promocao nova">Venha conversar conosco e conhecer
nossas ofertas de <strong>bolsas de estudos</strong>.</p>

   <p>Cursos:</p>
   <article>
     <p class="novo-curso">Medicina</p>
     <p class="curso">Engenharia de Computação</p>
     <p class="curso">Farmácia</p>
   </article>

   <p id="contato" class="final-pagina">Entre em contato:
11223344</p>

   <p id="rodape" class="final-pagina">Todos os direitos
reservados</p>
 </main>

</body>

</html>
```

Considere a necessidade de aumentar o padding e trocar a cor de fundo dos elementos que são cursos. Podemos aplicar, além dos recursos já vistos, **seletores descendentes**. No exemplo dado, definimos todo o conteúdo da página em um elemento main e os parágrafos que representam cursos estão agrupados em um article. Desejamos aumentar o padding e trocar a cor de fundo de todos os elementos p que sejam filhos (diretos ou "subfilhos", "subsubfilhos" e assim por diante) de um article. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Mais seletores</title>
 <!-- vamos especificar as regras CSS aqui, para simplificar
-->
 <style>
    /* esse é um seletor descentente, representado pelo espaço
em branco. pegamos todos os p's que sejam filhos diretos ou
indiretos de um article. */
    article p{
      padding: 12px;
      background-color: lightgray;
    }
 </style>
</head>
…
```

Observe o resultado. Somente os elementos p que são filhos de um article foram selecionados:



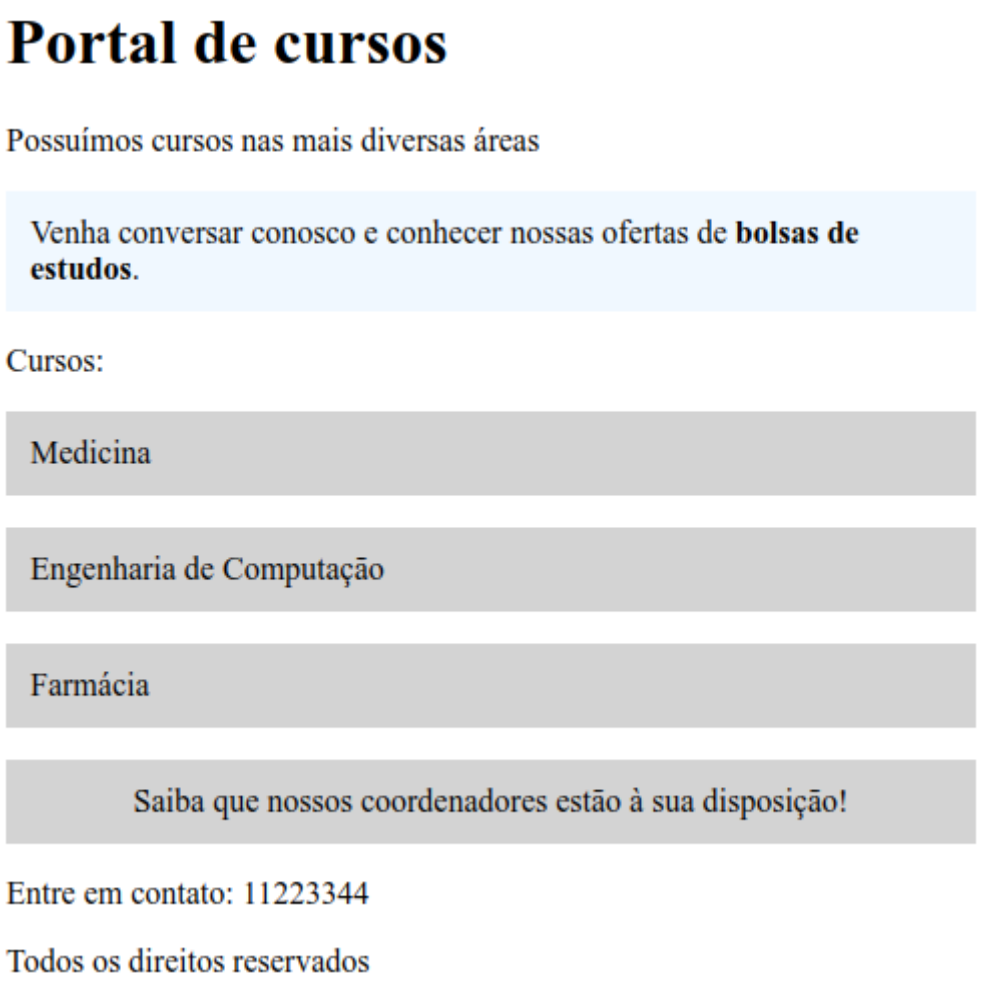
Faça a alteração a seguir e note que o novo parágrafo, embora não seja filho direto de article, também sofre a alteração o efeito das regras CSS:

```
...
<p>Cursos:</p>
   <article>
     <p class="novo-curso">Medicina</p>
     <p class="curso">Engenharia de Computação</p>
     <p class="curso">Farmácia</p>
     <section>
       <!-- esse p não é filho direto de article mas é
descendente. as regras css o atingem também -->
       <p class="coordenadores">
         Saiba que nossos coordenadores estão à sua
disposição!
       </p>
     </section>
   </article>
...
```

Seletores diferentes (por tipo, classe ou id) também podem ser “misturados”, construindo um seletor descendente mais complexo:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="utf-8">
 <title>Mais seletores</title>
 <!-- vamos especificar as regras CSS aqui, para simplificar
-->
 <style>
   /* esse é um seletor descentente, representado pelo espaço
em branco. pegamos todos os p's que sejam filhos diretos ou
indiretos de um article. */
   article p{
     padding: 12px;
     background-color: lightgray;
   }

   /* todo p que tenha a classe promoção e que seja
descendente de um elemento cujo id seja principal */
   #principal p.promocao {
     padding: 12px;
     background-color: aliceblue;
   }
 </style>
</head>

<body>
 <main id="principal">
   <h1 id="titulo">Portal de cursos</h1>
```

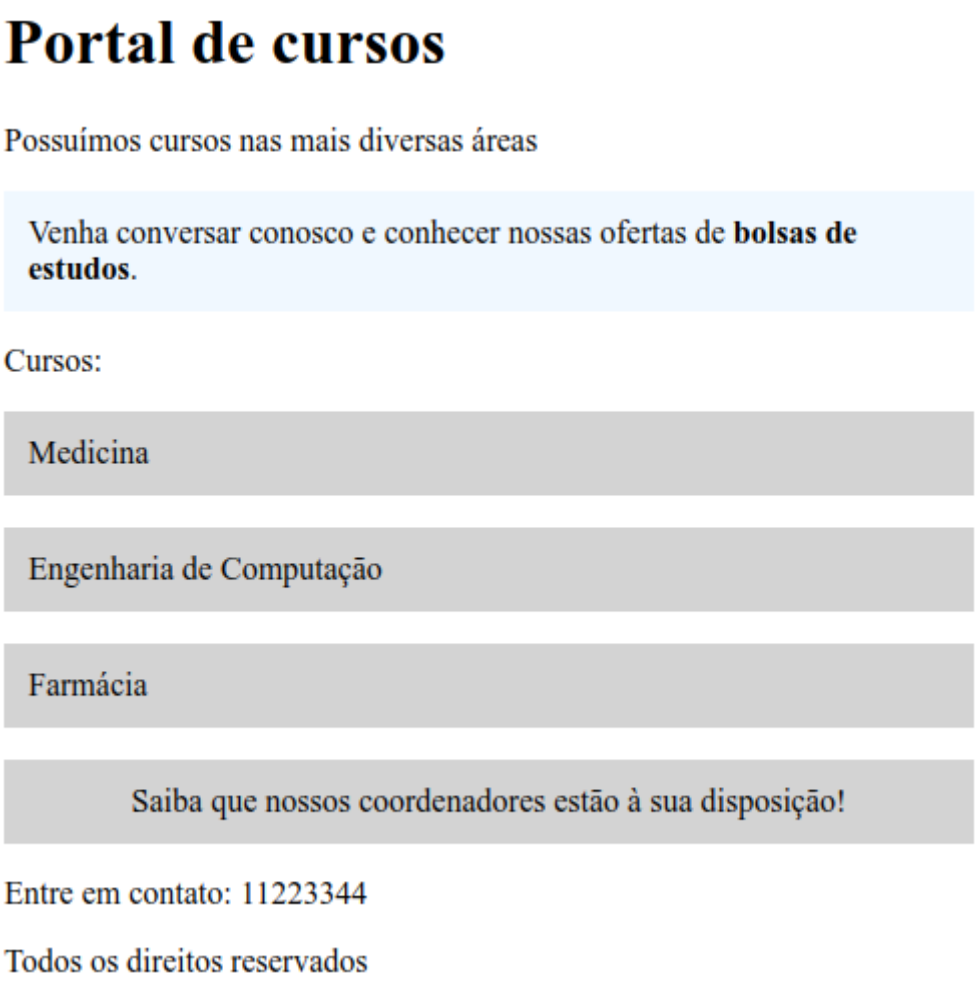
Resultado esperado:



Veja um exemplo ainda mais específico:

```
…
<head>
 <meta charset="utf-8">
 <title>Mais seletores</title>
 <!-- vamos especificar as regras CSS aqui, para simplificar
-->
 <style>
   /* esse é um seletor descentente, representado pelo espaço
em branco. pegamos todos os p's que sejam filhos diretos ou
indiretos de um article. */
   article p{
     padding: 12px;
     background-color: lightgray;
   }

   /* todo p que tenha a classe promoção e que seja
descendente de um elemento cujo id seja principal */
   #principal p.promocao {
     padding: 12px;
     background-color: aliceblue;
   }
   /* todo p que tenha a classe coordenadores, que seja
descendente de um section, que seja descendente de um
article, que seja descendente de um elemento de id igual a
principal, que seja descendente de body */
   body #principal article section p.coordenadores {
     text-align: center;
   }
 </style>
</head>
```

Observe que o texto sobre os coordenadores deve ter sido centralizado horizontalmente:

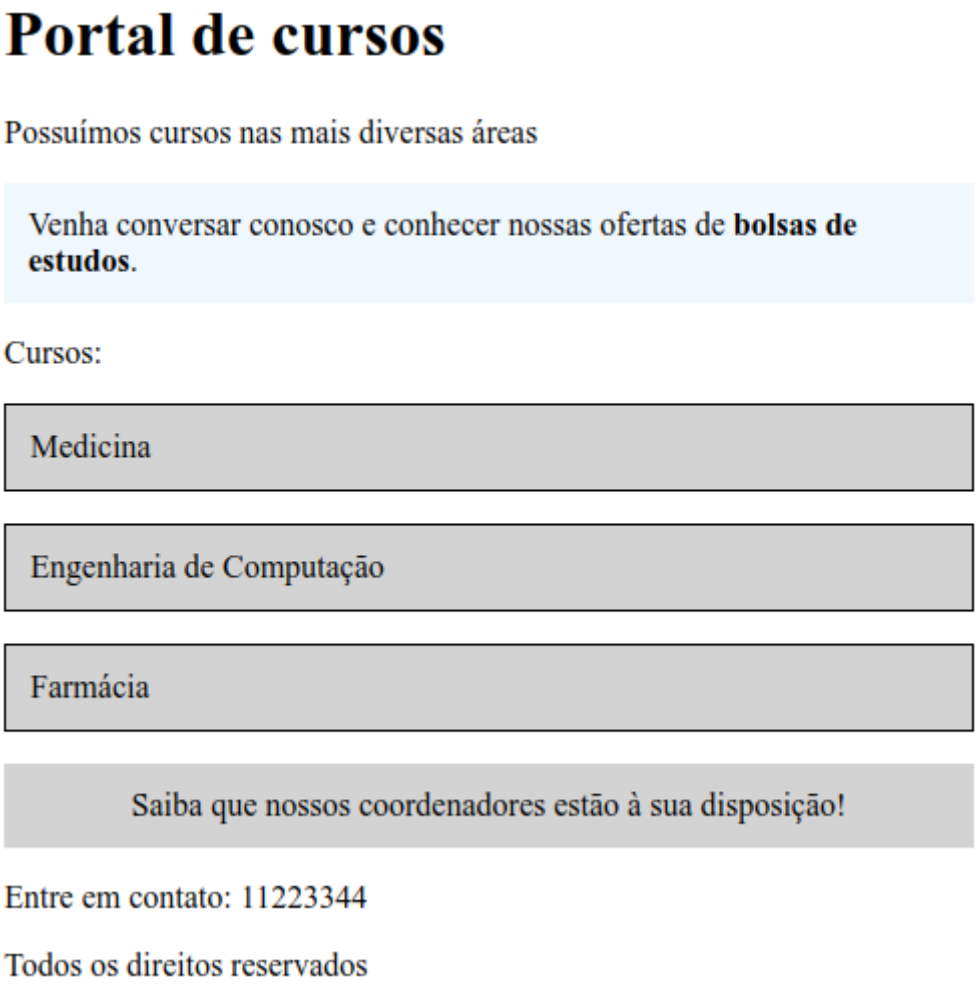


(Seletores de filhos diretos) O símbolo **>** permite selecionar elementos que sejam filhos diretos de um determinado elemento. Suponha que desejamos selecionar todos os parágrafos que sejam filhos diretos de article e somente eles. Podemos fazer assim:

```
...
<style>
   /* esse é um seletor descentente, representado pelo espaço em branco. pegamos todos os p's que sejam filhos diretos ou indiretos de um article. */
   article p{
     padding: 12px;
     background-color: lightgray;
   }

   /* todo p que tenha a classe promoção e que seja descendente de um elemento cujo id seja principal */
```

```
    #principal p.promocao {
      padding: 12px;
      background-color: aliceblue;
    }
    /* todo p que tenha a classe coordenadores, que seja
descendente de um section, que seja descendente de um
article, que seja descendente de um elemento de id igual a
principal, que seja descendente de body */
    body #principal article section p.coordenadores {
      text-align: center;
    }
    /* todos os p's que sejam filhos diretos de article. O p
que é filho de section não é selecionado, portanto. */
    article > p {
      border: 1px solid black
    }
  </style>

…
```

Veja o resultado visual. Observe que somente os cursos têm borda:

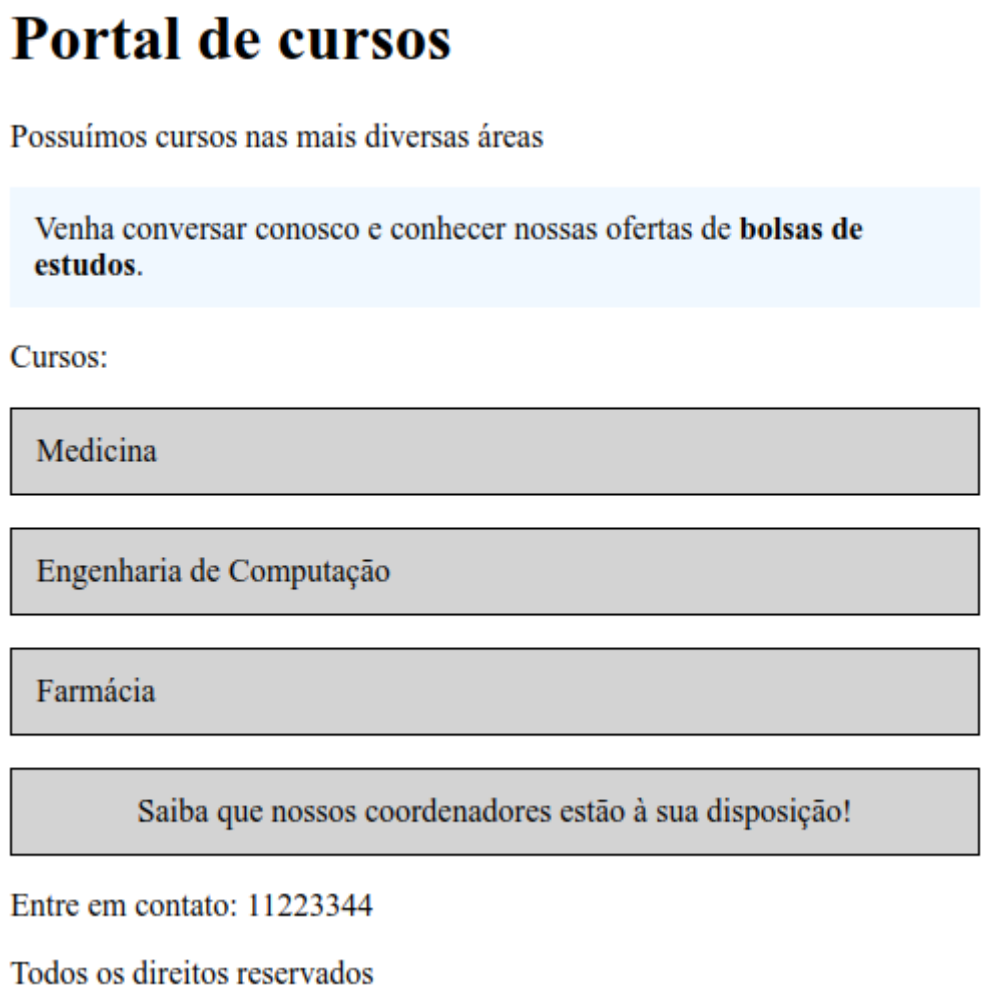


Agora remova o símbolo >, deixando um espaço em branco:

```
…
<style>
   /* esse é um seletor descentente, representado pelo espaço em branco. pegamos todos os p's que sejam filhos diretos ou indiretos de um article. */
   article p{
     padding: 12px;
     background-color: lightgray;
   }

   /* todo p que tenha a classe promoção e que seja descendente de um elemento cujo id seja principal */
   #principal p.promocao {
```

```
      padding: 12px;
      background-color: aliceblue;
    }
    /* todo p que tenha a classe coordenadores, que seja
descendente de um section, que seja descendente de um
article, que seja descendente de um elemento de id igual a
principal, que seja descendente de body */
    body #principal article section p.coordenadores {
      text-align: center;
    }
    /* agora é seletor descendente. o p que é filho de section
também é selecionado. */
    article  p {
      border: 1px solid black
    }
  </style>

…
```

Observe que a caixa sobre os coordenadores agora também tem borda:

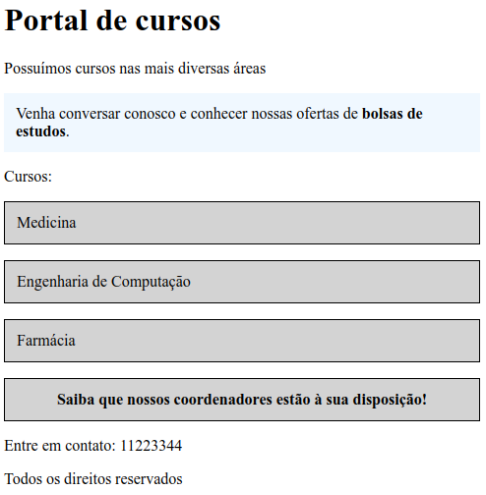


Veja um exemplo mais específico:

```
…
<style>
   /* esse é um seletor descentente, representado pelo espaço em branco. pegamos todos os p's que sejam filhos diretos ou indiretos de um article. */
   article p{
     padding: 12px;
     background-color: lightgray;
   }

   /* todo p que tenha a classe promoção e que seja descendente de um elemento cujo id seja principal */
   #principal p.promocao {
```

```
      padding: 12px;
      background-color: aliceblue;
    }
    /* todo p que tenha a classe coordenadores, que seja descendente de um section, que seja descendente de um article, que seja descendente de um elemento de id igual a principal, que seja descendente de body */
    body #principal article section p.coordenadores {
      text-align: center;
    }
    /* agora é seletor descendente. o p que é filho de section também é selecionado. */
    article  p {
      border: 1px solid black
    }
    /* todo p que tenha a classe coordenadores, que seja filho direto de um section, que seja descendente de um article, que seja filho direto de um main */
    main > article section > p.coordenadores {
      font-weight: bold;
    }
  </style>

...
```

Resultado:



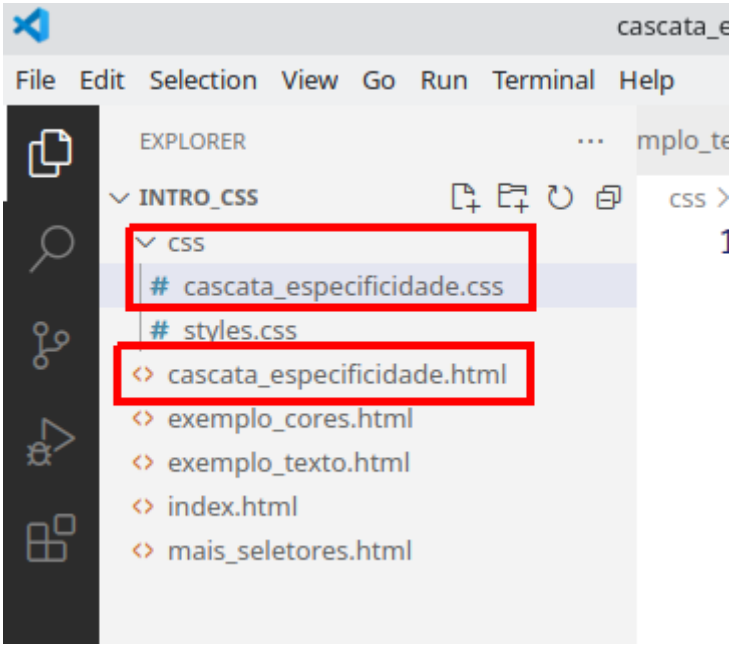
A tabela a seguir mostra um breve resumo do significado dos símbolos que podemos utilizar para especificar seletores CSS.

Símbolo	Significado	Exemplo	Funcionamento
espaço em branco	Seletor descendente, para filhos, "subfilhos", "subsubfilhos" etc	article .destaque	Seleciona todos os elementos que tenham destaque como classe e que sejam descendentes de um article.
vírgula	Agrupamento. Funciona como um **"ou lógico".**	.destaque, .promocao	Seleciona todos os elementos que tenham destaque **ou** promocao como classe.
>	Seletor de filhos diretos	.principal > section	Seleciona todos os elementos do tipo section que sejam filhos diretos de um elemento que tenha principal como classe.

(Efeito em cascata do CSS e especificidade) Uma pergunta natural a ser feita é a seguinte: caso existam duas regras conflitantes (uma delas troca a cor da fonte para azul e a outra para vermelho, por exemplo) operando sobre o mesmo elemento, qual delas prevalece? Para estudar esse tópico, crie um arquivo chamado **cascata_especificidade.html**. Veja seu conteúdo inicial:

```
!<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <title>Cascata e especificidade</title>
 </head>
 <body>

 </body>
</html>
```

Também precisaremos de um arquivo para abrigar regras CSS. Ele pode se chamar **cascata_especificidade.css** e ser criado na pasta css do projeto. Veja:



Não deixe de importar o arquivo CSS no arquivo HTML:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
 </head>
 <body>

 </body>
</html>
```

(Regra no arquivo .css) Considere a existência de um elemento p cuja cor alteramos escrevendo uma regra no arquivo CSS:

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

Digamos que ele seja definido da seguinte forma no arquivo HTML:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
 </head>
 <body>
   <p>Um parágrafo</p>
 </body>
</html>
```

Neste momento, claro, o texto deve estar vermelho:

(Regras conflitantes no arquivo .css e no elemento style (elemento style prevalece??)) A seguir, suponha que definamos uma regra CSS que altera a cor de todos os p's para azul, utilizando o elemento **style** do arquivo HTML:

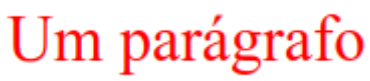
```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
      p {
        color: blue;
      }
   </style>
 </head>
 <body>
   <p>Um parágrafo</p>
 </body>
</html>
```

Observe que agora o texto ficou azul:

(Regras conflitantes no arquivo .css e no elemento style: alterando a ordem dos elementos link e style) O que o exemplo anterior sugere, é que regras CSS definidas no elemento style prevalecem sobre regras definidas em arquivos externos. Entretanto, façamos o seguinte teste: **definamos o elemento style antes de fazer a importação do arquivo externo usando o elemento link**. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
  <meta charset="utf-8">
  <style>
    p {
      color: blue;
    }
  </style>
  <link rel="stylesheet" href="css/cascata_especificidade.css">
  <title>Cascata e especificidade</title>
</head>
<body>
  <p>Um parágrafo</p>
</body>
</html>
```

O texto fica vermelho:



Isso ilustra parte do efeito cascata do CSS: **prevalece a regra definida por "último", considerando que o processamento do código se dá sempre de cima para baixo**.

(Regras conflitantes com CSS inline) Agora, atribuamos a cor verde ao elemento p utilizando CSS inline:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
      p {
        color: blue;
      }
   </style>
 </head>
 <body>
   <p style="color: green;">Um parágrafo</p>
 </body>
</html>
```

Observe que o texto fica verde:

Isso quer dizer que **regras definidas por meio de CSS inline prevalecem sobre regras definidas por meio do elemento style**.

Não faz muito sentido, mas podemos especificar uma segunda cor como CSS inline:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
     p {
       color: blue;
     }
   </style>
 </head>
 <body>
   <p style="color: green; color: yellow">Um parágrafo</p>
 </body>
</html>
```

Observe como o texto ficou amarelo:

Assim, **caso tenhamos duas regras definidas por meio de CSS inline conflitantes, a que prevalece é a última**.

Uma próxima pergunta a ser respondida é a seguinte: o que ocorre se tivermos duas regras conflitantes definidas no elemento **style**? Claro, para que o teste faça sentido, precisamos remover o CSS inline, caso contrário ele prevalecerá:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
     /* duas regras conflitantes */
     p {
       color: blue;
     }
```

```
      p{
        color: red;
      }
    </style>
  </head>
  <body>
    <!-- removemos o css inline -->
    <p>Um parágrafo</p>
  </body>
</html>
```

Neste caso, o texto fica vermelho:

Ou seja, **caso tenhamos regras CSS conflitantes definidas no elemento style, a que prevalece é aquela que aparece por último (mais ou menos, ainda precisamos falar sobre especificidade)**.

O mesmo vale para regras conflitantes definidas no arquivo .css: A que prevalece é a que aparece por último. No arquivo HTML, removemos as regras do elemento style para que as regras do arquivo tenham chance de causar efeito na página:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="css/cascata_especificidade.css">
    <title>Cascata e especificidade</title>
    <style>
      /* removemos as regras, assim aquelas definidas no arquivo
externo têm alguma chance */
    </style>
  </head>
  <body>
    <!-- removemos o css inline -->
    <p>Um parágrafo</p>
  </body>
</html>
```

No arquivo CSS, escrevemos duas regras conflitantes:

```
p {
 color: red;
}

p {
 color: yellow;
}
```

Observe que o texto fica amarelo:

Assim, estamos diante do **efeito em cascata do CSS**.

(O modificador !important) O modificador **!important** permite que o efeito em cascata do CSS seja cancelado. Para testá-lo, aplique uma cor via CSS inline mais uma vez ao elemento p:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
     /* removemos as regras, assim aquelas definidas no arquivo
externo têm alguma chance */
   </style>
 </head>
 <body>
   <!-- removemos o css inline -->
   <p style="color: blue">Um parágrafo</p>
 </body>
</html>
```

Apague todo o conteúdo do arquivo CSS e escreva somente uma regra:

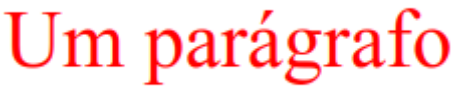
```
p {
  color: red;
}
```

São duas regras conflitantes e, pelo efeito em cascata do CSS, a que prevalece é aquela definida com CSS inline. O texto fica azul:

Mas disso já sabíamos. Ocorre que esta seção é sobre o modificador !important. Aplique-o da seguinte forma à regra CSS definida no arquivo CSS:

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

O texto fica vermelho:



Assim,

- Caso tenhamos diversas regras conflitantes, prevalece aquela definida pelo efeito em cascata do CSS.

- Além disso, o modificador !important cancela o efeito em cascata e, caso aplicado, passa a valer a regra a que ele foi aplicado.

- Caso o modificador de acesso !important seja aplicado a duas ou mais regras conflitantes, volta a valer o efeito cascata considerando apenas aquelas regras marcadas com !important.

Nota. O uso do modificador **!important** tende a tornar o processamento do CSS menos intuitivo. É recomendado evitar o seu uso.

(Algoritmo de especificidade) Além do efeito em cascata do CSS que estudamos, há um algoritmo chamado algoritmo de especificidade aplicado pelos navegadores para decidir, entre uma coleção de regras conflitantes, qual delas prevalece. Ele funciona da seguinte forma:

- para cada seletor, um peso é calculado
- o peso de um seletor é um valor de três colunas baseado em categorias
- as categorias são: id, classe e tipo
- para cada categoria existente no seletor, algoritmo faz a seguinte soma
- se a categoria for **id**, adicione 1-0-0 ao peso do seletor
- se a categoria for **class**, adicione 0-1-0 ao seletor
- se a categoria for **tipo**, adicione 0-0-1 ao seletor

Passemos aos nossos testes. Ajuste o conteúdo do arquivo **cascata_especificidade.html** da seguinte forma:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>

   </style>
 </head>
 <body>
   <main>
     <article >
       <p id="titulo" class="descricao">
         Algoritmo de especificidade do CSS
       </p>
     </article>
   </main>
 </body>
</html>
```

A seguir, faça a especificação das seguintes regras CSS, na ordem que aparecem. Vamos utilizar apenas o arquivo HTML neste exemplo. Certifique-se de remover todas as regras definidas no arquivo CSS, elas podem atrapalhar agora.

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
     /*especifidade: 1-0-0  */
     /* dá para pensar que é uma centena de pontos */
     #titulo {
       color: blue;
     }
     /* especificidade: 0-1-0 */
     /* dá para pensar que é uma dezena de pontos */
     .descricao{
       color: green;
     }
     /* especificidade: 0-0-1 */
     /* dá para pensar que é um único ponto */
     p{
       color: red;
     }
   </style>
 </head>
 <body>
   <main>
     <article >
       <p id="titulo" class="descricao">
         Algoritmo de especificidade do CSS
       </p>
     </article>
   </main>
 </body>
</html>
```

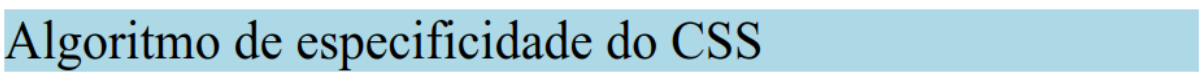
O texto fica azul:

Faça alguns testes alterando a ordem dos seletores. Observe que o texto permanece azul. Isso acontece pois o **seletor por id tem maior especificidade, ele é mais específico**.

A seguir, faça os seguintes ajustes no HTML:

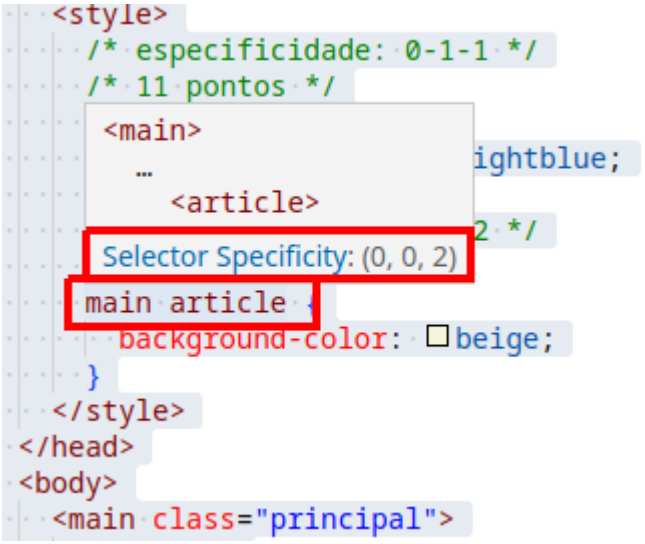
```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
     /* especificidade: 0-1-1 */
     /* 11 pontos */
     .principal article{
       background-color: lightblue;
     }
     /* especificidade: 0-0-2 */
     /* 2 pontos */
     main article {
       background-color: beige;
     }
   </style>
 </head>
 <body>
   <main class="principal">
     <article>
       <p id="titulo" class="descricao">
         Algoritmo de especificidade do CSS
       </p>
     </article>
   </main>
 </body>
</html>
```

O fundo fica azul:



A regra que prevaleceu foi aquela que envolve uma classe e um tipo (11 pontos). A outra envolve dois tipos (2 pontos) e tem menor especificidade.

Dica. Pause o mouse em cima de um seletor e veja a sua especificidade exibidade pelo VS Code. Exemplo:



Observe que, quando utilizamos o operador vírgula, temos dois seletores, cada qual com a sua especificidade:

```
<!DOCTYPE html>
<html lang="pt-BR">
 <head>
   <meta charset="utf-8">
   <link rel="stylesheet" href="css/cascata_especificidade.css">
   <title>Cascata e especificidade</title>
   <style>
     /* main é um seletor e article é outro. main tem especificidade 1 (pois tem só um
tipo) e article tem especificidade 1 (pois também tem somente um tipo) */
     main, article {
       border: 1px solid black;
     }
   </style>
 </head>
 <body>
   <main class="principal">
     <article>
       <p id="titulo" class="descricao">
         Algoritmo de especificidade do CSS
       </p>
     </article>
   </main>
 </body>
</html>
```

(Cores) Há diferentes formas para se especificar cores em CSS. Nesta seção estudaremos algumas delas. O código HTML a ser usado de exemplo é o seguinte. Crie um arquivo chamado **exemplo_cores.html** para abrigá-lo.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
   <meta charset="UTF-8">
   <title>Testando cores</title>
   <style>
        p{
           padding: 12px;
       }
   </style>
</head>
<body>
   <article id="cursos">

       <p class="engenharia">
           Engenharia de Computação
       </p>

       <p class="farmacia">
           Farmácia
       </p>

       <p class="biologia">
           Biologia
       </p>

       <p class="matematica">
           Matemática
       </p>

     </article>
</body>
</html>
```

(Nomes de cores pré-definidos) Como já vimos, uma das formas de se especificar cores é por meio de **nomes pré-definidos**. Eles fazem parte da especificação da linguagem CSS. Veja a seguir. Faça as definições CSS no elemento style do próprio arquivo html, para simplificar o exemplo.

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
   <meta charset="UTF-8">
   <title>Testando cores</title>
   <style>
        p{
           padding: 12px;
        }
        .engenharia {
          background-color: green;
          color: azure;
        }
   </style>
</head>
…
```

(RGB) Outra possibilidade é fazer a especificação da cor desejada utilizando o sistema de cores RGB. Utilizamos valores numéricos entre 0 e 255 (ou seja, 8 bits por cor) para especificar a "quantidade" de cor desejada para cada uma das que compõem o nome do sistema: **R**ed, **G**reen e **B**lue. Veja um exemplo:

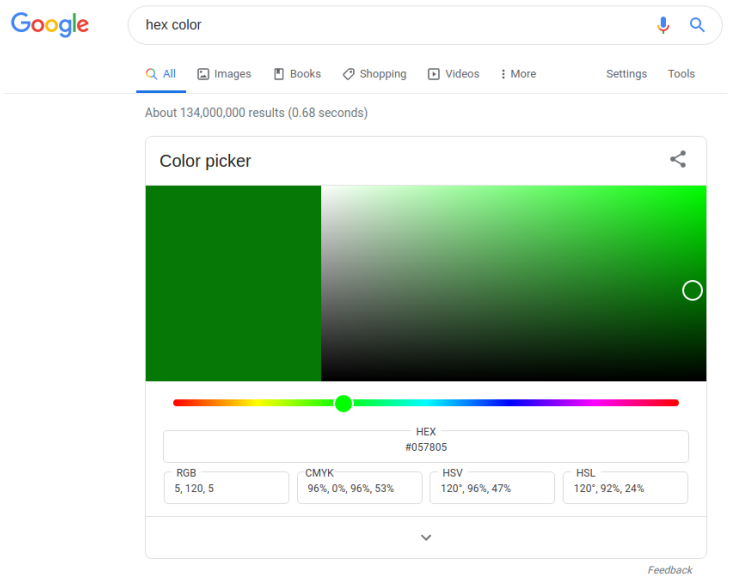
```
…
<style>
    p {
      padding: 12px;
    }

    .engenharia {
      background-color: green;
      color: azure;
    }

    .farmacia {
      /* cuidado para não colocar um espaço em branco entre rgb e o símbolo ( não funciona */
      color: rgb(255, 0, 0);
      /* observe que valores iguais para as três cores sempre resultam num tom de cinza. Quanto mais próximo de 0, mais próximo de preto. Quanto mais próximo de 255, mais próximo de branco. */
      background-color: rgb(240, 240, 240);
    }
  </style>

…
```

(Cores RGB por meio de números na base hexadecimal) Também é possível especificar cores usando o sistema de cores RGB sem utilizar a função rgb. Isso é feito por meio do uso de números na base hexadecimal. A notação envolvida inclui o símbolo #. Observe um exemplo:

```
…
<style>
    p {
      padding: 12px;
    }

    .engenharia {
      background-color: green;
```

```
      color: azure;
    }

    .farmacia {
      /* cuidado para não colocar um espaço em branco entre
rgb e o símbolo ( não funciona */
      color: rgb(255, 0, 0);
      /* observe que valores iguais para as três cores sempre
resultam num tom de cinza. Quanto mais próximo de 0, mais
próximo de preto. Quanto mais próximo de 255, mais próximo de
branco. */
      background-color: rgb(240, 240, 240);
    }

    .biologia {
      color: #057805;
      background-color: #00FF00;
    }
  </style>

...
```

Dica. Busque por “hex color” na omnibox (a barra de busca) do Google Chrome para encontrar a seguinte ferramenta:



Observe que ela permite que você faça a escolha das cores visualmente e lhe entrega os códigos hexadecimais referentes a elas.

(Cores RGB por meio de números na base hexadecimal: um atalho) No exemplo anterior, utilizamos 2 caracteres em base hexadecimal (de 0 a F) para representar cada uma das cores do sistema RGB. Também é possível fazer a especificação usando um único caractere por cor. Quando isso é feito, entende-se que o segundo caractere de cada cor é igual ao primeiro. Veja:

```
…
<style>
   p {
     padding: 12px;
   }

   .engenharia {
     background-color: green;
     color: azure;
   }

   .farmacia {
     /* cuidado para não colocar um espaço em branco entre rgb e o símbolo
( não funciona */
     color: rgb(255, 0, 0);
     /* observe que valores iguais para as três cores sempre resultam num
tom de cinza. Quanto mais próximo de 0, mais próximo de preto. Quanto mais
próximo de 255, mais próximo de branco. */
     background-color: rgb(240, 240, 240);
   }

   .biologia {
     color: #057805;
     background-color: #00FF00;
   }

   .matematica {
     background-color: #CCC;
     color: #000;
   }
 </style>

…
```

(Formatação de texto) Nesta seção, passamos a estudar a formatação de texto com CSS. Para isso, crie um novo arquivo chamado **exemplo_texto.html**. Seu conteúdo inicial é o seguinte:

```
<!DOCTYPE html>
<html lang="pt-BR">
<head>
 <meta charset="UTF-8">
 <title>Formatando texto</title>
 <style>

 </style>
</head>

<body>
 <main>
   <article id="cursos">

     <p class="engenharia">
       Engenharia de Computação
     </p>

     <p class="farmacia">
       Farmácia
     </p>
   </article>
 </main>
</body>
</html>
```

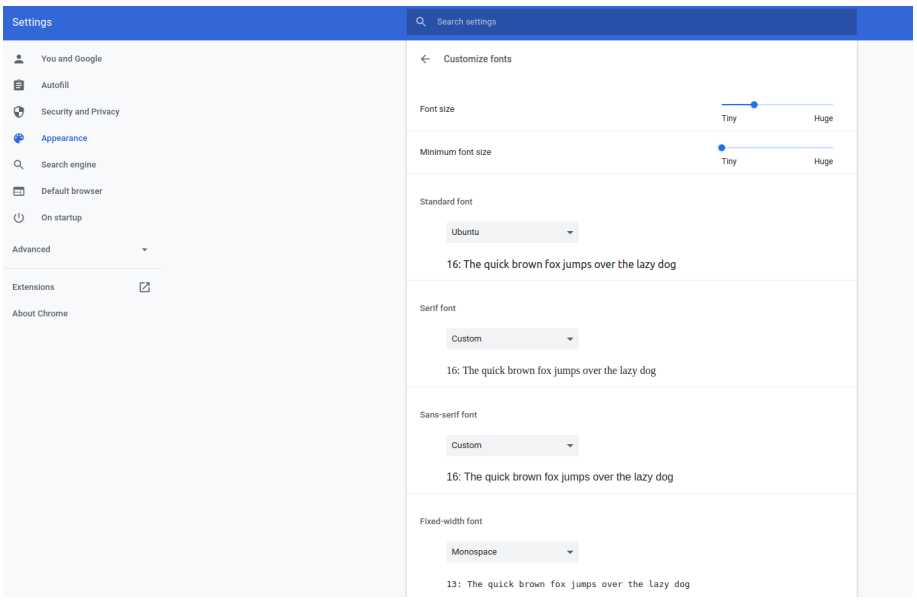
Começamos alterando a sua cor (como já vimos), a fonte usada e o tamanho dela:

```
…
<style>
    .engenharia {
      color: #0000FF;
      font-family: 'Courier New';
      font-size: 20px;
    }
 </style>

…
```

Note que, para a fonte, especificamos uma “família” de fontes. Neste caso, o navegador irá utilizar a primeira, da esquerda para a direita, que estiver disponível no computador. Veja mais um exemplo:

```
…
<style>
    .engenharia {
      color: #0000FF;
      font-family: 'Courier New', Courier, monospace
    }
 </style>

…
```

(Fontes padrão do navegador e serifas) A fonte sendo exibida depende do navegador e do sistema operacional sendo utilizados. Cada navegador pode especificar as suas próprias fontes padrão e cada sistema operacional pode ter seu próprio conjunto de fontes instalado. No Google Chrome, você pode verificar as suas fontes padrão clicando nos **três pontinhos no canto superior direito** e então: **Configurações >> Aparência >> Personalizar Fontes**. Veja:



Observe que o navegador define:

- Uma fonte padrão
- Uma fonte com serifa
- Uma fonte sem serifa
- Uma fonte de largura fixa

Nota. Serifas são os traços que ocorrem no fim das hastes das letras. A figura a seguir destaca as serifas da letra T usando a fonte **Times New Roman**, que é uma fonte que tem serifa.

A figura a seguir mostra a letra T usando a fonte **Helvetica,** que é uma fonte sem serifa.

Temos algumas opções para especificar a fonte a ser utilizada pelo navegador:

- **sans-serif**: instruímos o navegador a usar a sua fonte sem serifa padrão
- **serif**: instruímos o navegador a usar a sua fonte com serifa padrão
- **monospace**: instruímos o navegador a usar a sua fonte de largura fixa padrão
- **nome**: especificamos o nome da fonte que o navegador deve utilizar

Faça, um a um, os testes exibidos a seguir. Observe como especificamos uma "família de fontes". A primeira fonte especificada será utilizada caso esteja disponível no sistema operacional. Caso contrário, a segunda será utilizada. E assim por diante. Por isso é importante usar as palavras sans-serif, serif ou monospace como última opção. Se desejamos uma fonte sem serifa, especificamos a família desejada e se nenhuma estiver disponível, a especificação sans-serif instrui o navegador a usar a fonte padrão. E assim por diante.

```
…
<style>
    .engenharia {
      color: #0000FF;
      /* família de fontes com serifa */
      font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif
    }
 </style>
…
```

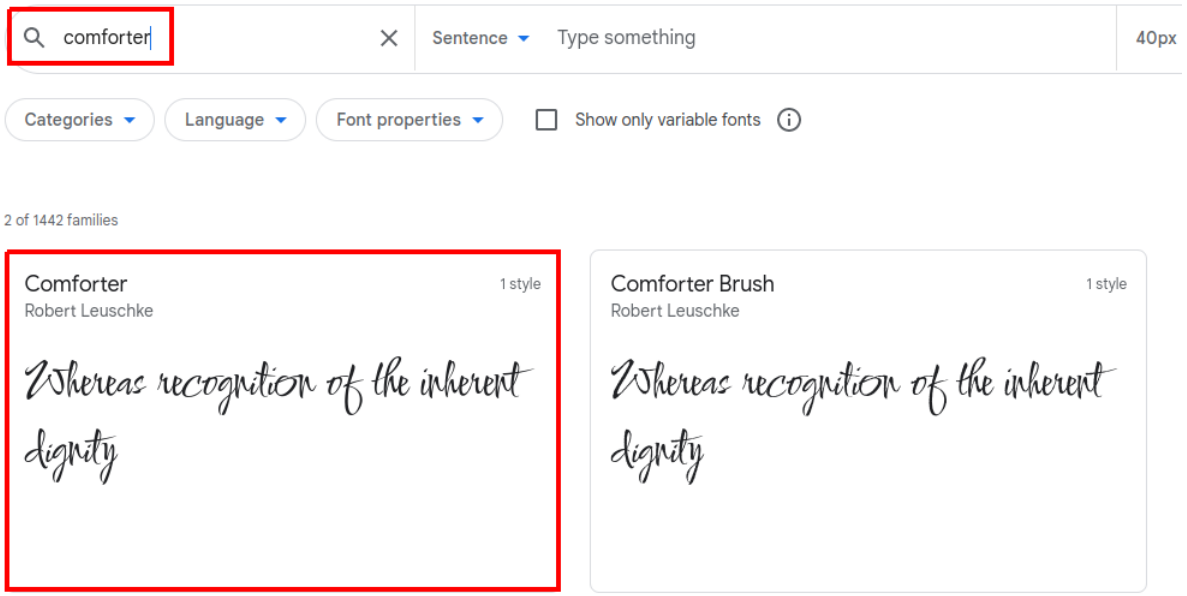
```
…
<style>
    .engenharia {
      color: #0000FF;
      /* família de fontes sem serifa */
      font-family: Arial, Helvetica, sans-serif
    }
 </style>
…
```

```
…
<style>
    .engenharia {
      color: #0000FF;
      /* família de fontes de largura fixa*/
      font-family: 'Courier New', Courier, monospace
    }
 </style

…
```

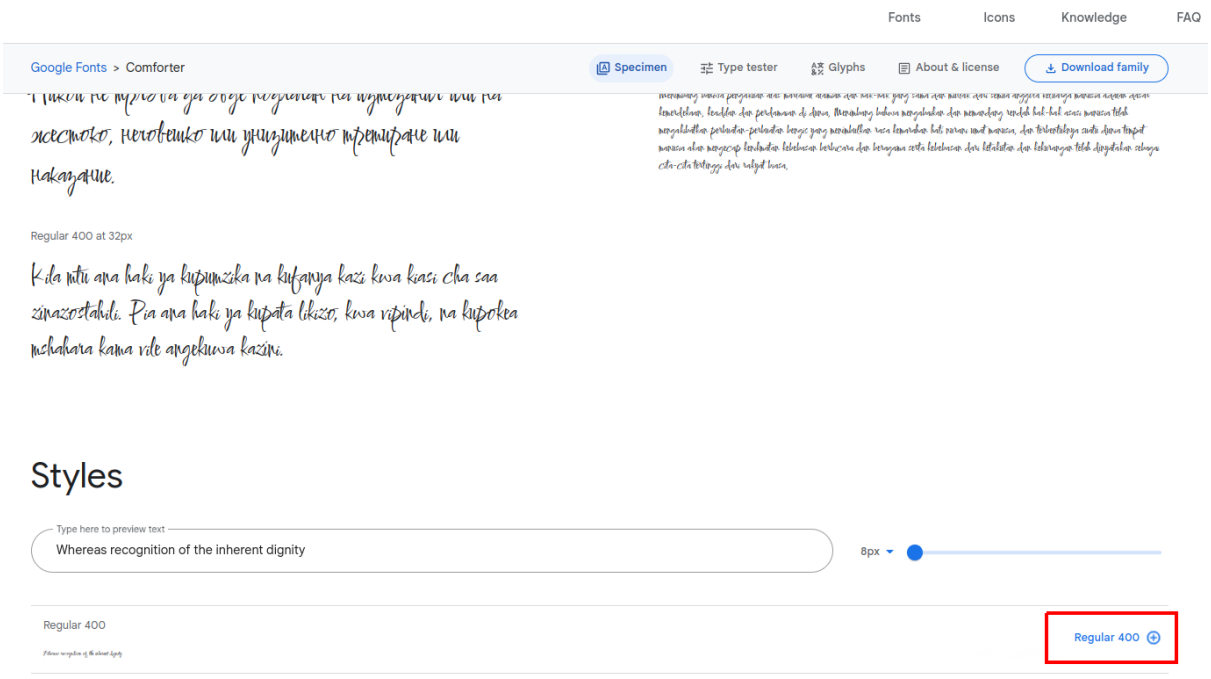
(Fontes Google) Podemos também fazer uso de fontes que não se encontram instaladas no sistema operacional em uso. Isso pode ser feito, por exemplo, utilizando-se as fontes Google. Para tal, comece visitando o link a seguir:

https://fonts.google.com/

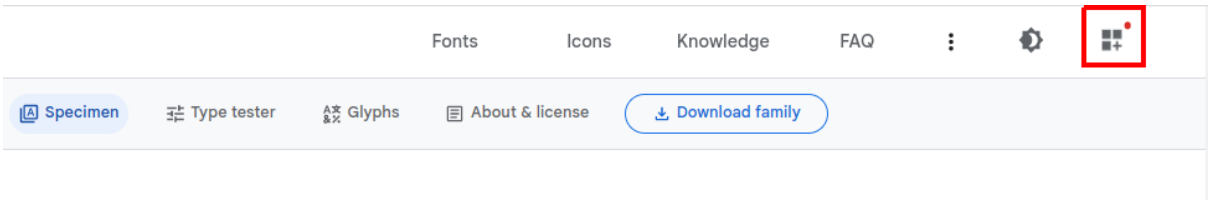
Escolha uma das fontes disponíveis. Neste exemplo, escolhemos a fonte chamada **Comforter**. Caso ela não esteja aparecendo na tela inicial, basta buscar pelo seu nome:



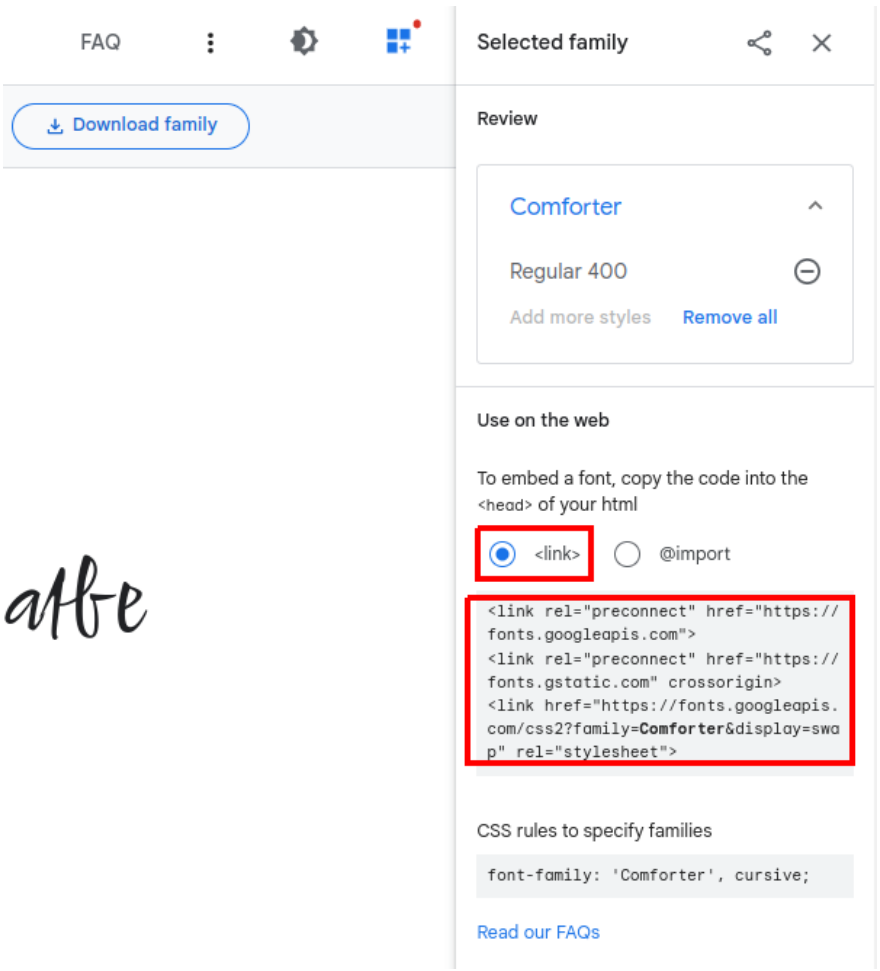
Depois de encontrá-la, clique sobre o cartão que exibe seu nome e uma amostra. Role a tela para baixo até encontrar a opção **Regular 400**. Observe que ela tem um sinal de + do lado:



A seguir, deve ser possível clicar no botão de fontes selecionadas no canto superior direito da tela. Veja:



Para usar a fonte, basta manter a opção link selecionada e copiar o código que aparece abaixo dela:



O código que você copiou precisa ser colado na seção **head** do seu documento HTML. A seguir, basta associar a família de fontes à propriedade do elemento desejado:

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
 <meta charset="UTF-8">
 <link rel="preconnect" href="https://fonts.googleapis.com">
 <link rel="preconnect" href="https://fonts.gstatic.com"
crossorigin>
 <link
href="https://fonts.googleapis.com/css2?family=Comforter&disp
lay=swap" rel="stylesheet">
 <title>Formatando texto</title>
 <style>
```

```
    .engenharia {
      color: #0000FF;
      /* família de fontes de largura fixa*/
      font-family: 'Comforter', cursive;
    }
  </style>
</head>

<body>
  <main>
    <article id="cursos">

      <p class="engenharia">
        Engenharia de Computação
      </p>

      <p class="farmacia">
        Farmácia
      </p>
    </article>
  </main>
</body>

</html>
```

Nota. **cursive** é uma palavra genérica que estamos utilizando para instruir o navegador a utilizar uma fonte “manuscrita”, em que as letras são ligadas umas às outras, caso a fonte Comforter não seja encontrada.

Veja o resultado esperado:

Há diversas propriedades relativas à fonte a ser utilizada. Veja:

```
…
<style>
    .engenharia {
      color: #0000FF;
      /* família de fontes de largura fixa*/
      font-family: 'Comforter', cursive;
    }
    .farmacia{
      /* família de fontes */
      font-family: Arial, Helvetica, sans-serif;
      /* tamanho */
      font-size: 20px;
      /* itálico */
      font-style: italic;
      /* negrito */
      font-weight: bold;
    }
 </style>
…
```

Veja o resultado esperado:

Há um atalho para especificar todas as propriedades referentes à fonte de uma única vez:

```
…
.engenharia {
      color: #0000FF;
      /* família de fontes de largura fixa*/
      font-family: 'Comforter', cursive;
    }
    .farmacia{
      font: italic bold 20px Arial, Helvetica, sans-serif;
    }

…
```

Dica. Verifique a documentação do atributo font para entender a ordem em que os valores devem ser especificados. Se você pausar o mouse em cima da palavra font no VS Code, ele também mostrará as regras. Veja:

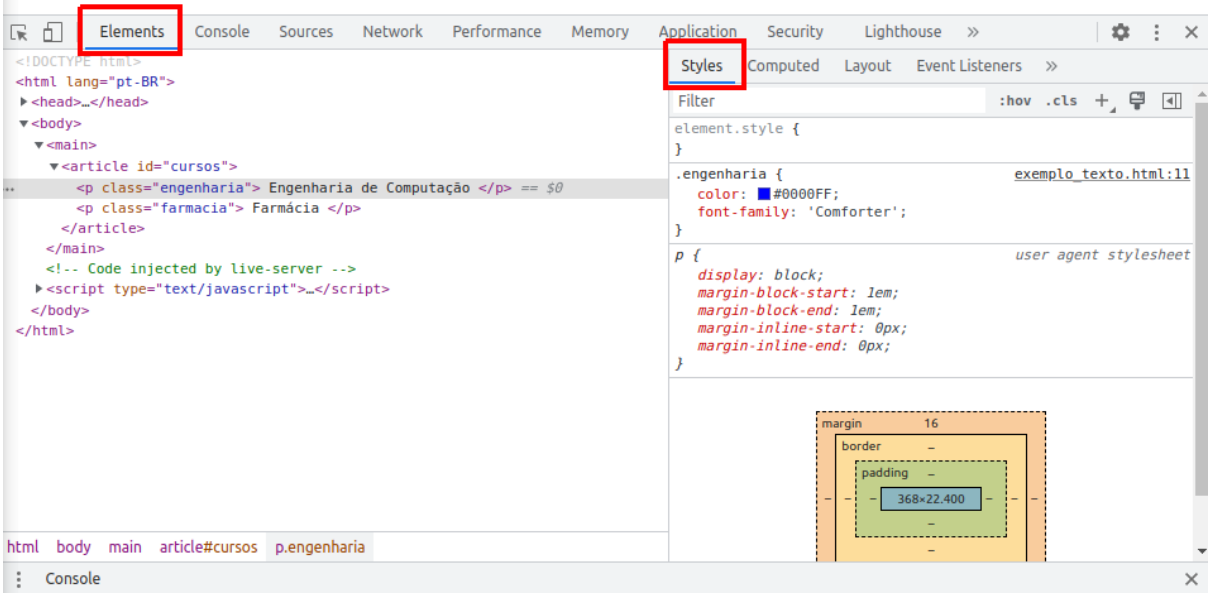
(text-decoration: sublinhando, riscando etc) Podemos riscar ou sublinhar (entre muitas outras coisas) o texto com a propriedade **text-decoration:**

```
…
<style>
    .engenharia {
      color: #0000FF;
      /* família de fontes de largura fixa*/
      font-family: 'Comforter', cursive;
    }
    .farmacia{
      font: italic bold 20px Arial, Helvetica, sans-serif;
      /* teste outros valores */
      text-decoration: line-through;
    }
 </style>
…
```

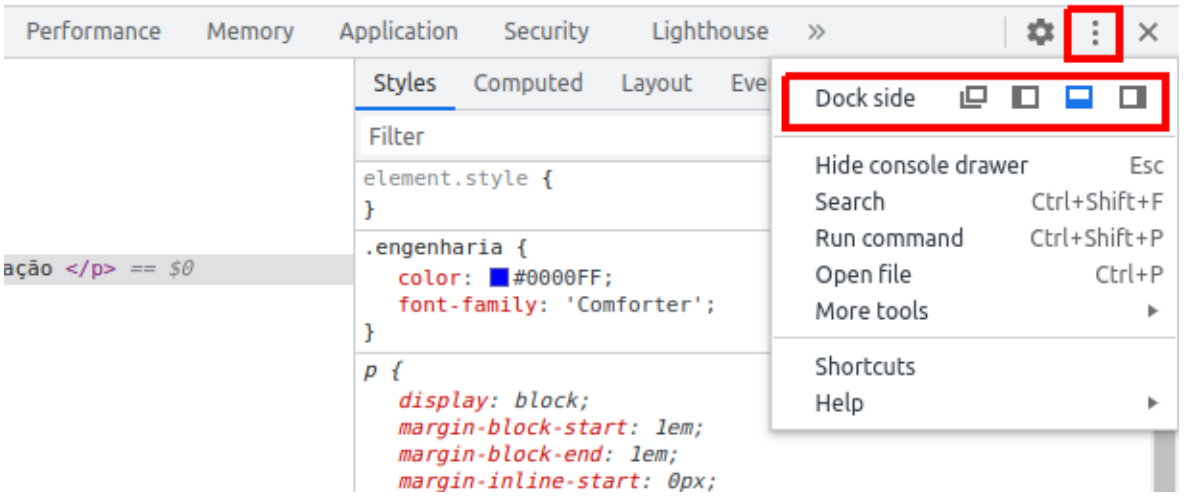
(Chrome Dev Tools) No Google Chrome, aperte CTRL + Shift + I para abrir o Chrome DevTools (https://developers.google.com/web/tools/chrome-devtools). Ele permite inspecionar nossos elementos e mesmo alterar algumas de suas propriedades. Veja a seguir. Vá até a aba **Elements** e então escolha a aba **Styles**.

Engenharia de Computação

~~**Farmácia**~~



Nota. O Chrome Dev Tools pode ser posicionado de qualquer lado da tela, abaixo da tela ou, ainda, separado dela. Para fazer a sua escolha, basta clicar nos três pontinhos no canto superior direito e usar uma das opções de **Dock Side**:



Usando o Chrome Dev Tools, estando em **Elements** e **Styles**, selecione um dos elementos p. A seguir, clique no nome da fonte que está atribuída a ele e troque para algo como Arial. Veja:

Engenharia de Computação

~~***Farmácia***~~