



APOSTILA DE HTML E CSS

PROFESSORA BIANCA MELO

SUMÁRIO

INTRODUÇÃO	1
HTML	1
Tags essenciais	1
Elementos da página	2
Listas	4
Tabelas	5
Âncoras	6
Links	6
Imagens.....	7
Referências.....	7
Formulários	7
Tipos de campo (input).....	8
Outros tipos de campo (não input)	9
Botões	10
CSS	11
Seletores	12
Comentários.....	13
Cores	13
Planos de fundo.....	14
Margens	15
Padding	16
Tamanho	16
Formatação de texto	16
Links	17
Posicionar elementos	18
Box Model	18
ACENTUAÇÃO E CARACTERES ESPECIAIS	19

INTRODUÇÃO

HTML é a sigla "HyperText Markup Language" (Linguagem de Marcação de Hipertexto). Lembre-se de que HTML não é uma linguagem de programação, e sim de marcação de conteúdo na web. O desenvolvimento de páginas web pode ser contemplado em três camadas: HTML, CSS e a programação de fato.

HTML - camada responsável pela marcação da estrutura da página. O HTML é a parte que vai marcar o conteúdo da página em colunas, parágrafos, cabeçalhos, formulários, entre outros.

CSS - camada responsável pela apresentação visual da página web. CSS é a abreviação de Cascade Style Sheet, ou Folha de Estilo em Cascata. O CSS é responsável por alterar a cor, fonte, tamanho e alinhamento dos textos e imagens. O CSS também é responsável pelo posicionamento das colunas. Tudo referente ao estilo da página é tratado pelo CSS.

Programação - a programação é a camada que dita o comportamento da página. Essa programação não é feita em HTML ou em CSS, mas sim em linguagens próprias como PHP ou Javascript.

HTML

Tags essenciais

Para que os elementos da página sejam corretamente marcados, é necessário usar tags. As tags estarão sempre entre < e >.

Todas as tags HTML e seus respectivos atributos podem ser consultados no site da W3C: www.w3c.br ou www.w3schools.com

Ao montarmos uma página web, é necessário ter, no mínimo, a seguinte estrutura:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> Título da página </title>
  Informações de estilo, scripts etc.
</head>
<body>
  Conteúdo que será exibido na página
</body>
</html>
```

A declaração **<!DOCTYPE html>** é utilizada para informar ao navegador (browser) que o documento renderizado é um documento HTML. Todo documento HTML deve iniciar com a declaração DOCTYPE para ser compatível com os padrões definidos pela W3C.

A tag **<html>** sempre terá que existir no início de uma página, para indicar ao browser qual a linguagem que está sendo utilizada para montar a página. Ao final da página, é necessário indicar "fechando" a tag, ou seja, usando **</html>**

A tag **<head>** contém informações sobre o documento, como o título que aparecerá na barra de títulos(que fica dentro das tags **<title>** e **</title>**), as informações de estilo e scripts que serão carregados na página.

Além disso, dentro da tag **<head>** também ficam tags **<meta>**, que descrevem os metadados de um documento HTML. No exemplo, temos uma tag meta definindo o **charset** (*character set*), conjunto de caracteres mapeados para algum código. Uma codificação (**encoding**) descreve como esses códigos são representados em bytes. Por exemplo, o UTF-8 possui quase todos os caracteres mapeados. Existem outros padrões, mas normalmente utilizamos esse.

A tag **<body>** compreende tudo que será mostrado na janela principal do browser, ou seja, a página de fato. Dentro do body, temos os cabeçalhos, parágrafos, colunas, listas, tabelas, links e imagens.

Para escrever comentários no arquivo HTML, devemos utilizar a seguinte estrutura: **<!-- comentário -->**. O texto entre **<!--** e **-->** não será visível na página.

Elementos da página

Essas são as tags que ficam dentro do **<body>**, e que marcam o que efetivamente será exibido no navegador.

Os **cabeçalhos** são utilizados para títulos e subtítulos de uma página. Note que o título da página que aparece dentro da tag **<title>** não precisa ser o mesmo que aparece no cabeçalho.

O HTML possui seis níveis de cabeçalhos, numerados de 1 a 6. O número 1 é o de maior destaque e, de acordo com as regras da W3C, deve ser único em cada página. O primeiro cabeçalho em cada página deve estar marcado como **<h1>**. Todos os outros níveis de cabeçalho podem aparecer inúmeras vezes dentro de cada página.

O conjunto de notações possíveis de cabeçalhos é o seguinte:

<pre><h1> Cabeçalho de nível 1 </h1> <h2> Cabeçalho de nível 2 </h2> <h3> Cabeçalho de nível 3 </h3> <h4> Cabeçalho de nível 4 </h4> <h5> Cabeçalho de nível 5 </h5> <h6> Cabeçalho de nível 6 </h6></pre>
--

Não é possível colocar um cabeçalho dentro do outro. Os cabeçalhos podem ficar dentro de colunas, mas não dentro de parágrafos.

Os **separadores** são elementos de grande importância no HTML, já que a linguagem não reconhece o fim de uma linha ou parágrafo ao pressionar a tecla Enter. Se você simplesmente

escrever palavras sem tags de formatação em um arquivo HTML, elas serão formatadas em um grande parágrafo sem quebras de linha. Portanto, é necessário forçar as quebras de linha e a criação dos parágrafos utilizando os separadores.

A tag **<div>** indica o início de uma divisão na página, ou uma coluna. Essa tag sempre deve ser fechada com **</div>**, e pode conter outras divs e parágrafos. As divs são muito importantes na construção de páginas, já que é por meio delas que demarcamos o posicionamento dos elementos nas páginas, e é por isso que, ao entrarmos em alguns sites, vemos diferentes "blocos" de texto posicionados em lugares específicos, alguns lado a lado, outros sempre acima da página etc. Sem as divs, esses blocos apareceriam sempre do mesmo jeito: um após o outro, conforme rolamos a página.

A tag **<p>** iniciará um novo parágrafo. É importante marcar também o fim do parágrafo, fechando a tag com **</p>**. Um parágrafo não deve conter outros parágrafos.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title> Exemplo - parágrafos e DIVs </title>
</head>
<body>
<h1>Cabeçalho principal</h1>
<div id="principal">
Esta é a div principal, que contém todos os
outros elementos da página.

<p class="texto">Este é um parágrafo.</p>
<p class="texto">Este é outro parágrafo.</p>

<div id="interno">Esta é uma div interna, que
fica dentro da div principal</div>
</div>
</body>
</html>
```

Os **atributos** **class** e **id**, utilizados dentro desses elementos, servem para demarcá-los ou diferenciá-los conforme uma categoria. O atributo **id** é um identificador e, portanto, deve ser sempre único. O atributo **class** serve para definir características comuns a vários elementos. A mesma tag pode ter atributos **class** e **id** ao mesmo tempo.

A tag **
** é a responsável por quebras de linha dentro dos parágrafos. A tag BR não possui fechamento, pois ela apenas marca a quebra da linha. Caso você queira pular duas linhas, insira a tag BR duas vezes, assim:

Efeito na página	Código HTML
Texto 1	Texto 1
Texto 2	Texto 2
Texto 3	Texto 3

Listas

As **listas** demarcam elementos com estrutura de listas. Há dois tipos de listas: listas numeradas ou ordenadas e listas não ordenadas.

As **listas ordenadas** são demarcadas pelas tags e (OL = ordered list = lista ordenada), e cada item da lista é demarcado pelas tags e (LI = list item = item da lista).

Efeito na página	Código HTML
1. Arroz	
2. Feijão	Arroz
3. Salsicha	Feijão
4. Batata palha	Salsicha
	Batata palha
	

As listas ordenadas podem possuir atributos, como o **TYPE**. O atributo type especifica o tipo de marcador que será utilizado na lista (letras, números, algarismos romanos etc)

Valor do atributo TYPE	Descrição
1	Organiza a lista em algarismos arábicos - padrão do HTML Cada item será identificado por 1, 2, 3 etc
I	Organiza a lista em algarismos romanos maiúsculos Cada item será identificado por I, II, III etc
i	Organiza a lista em algarismos romanos minúsculos Cada item será identificado por i, ii, iii etc
A	Organiza a lista em ordem alfabética maiúscula Cada item será identificado por A, B, C etc
a	Organiza a lista em ordem alfabética minúscula Cada item será identificado por a, b, c etc

Outros atributos que as listas ordenadas também possuem: **REVERSED**, que especifica que a lista deve estar em ordem decrescente, e **START**, que define o valor de início. Esses atributos podem ou não ser combinados. *O atributo reversed é novo, criado para a versão 5 do HTML, e pode não funcionar em alguns navegadores.*

Exemplo

Para criar uma lista ordenada organizada em algarismos romanos maiúsculos, em ordem decrescente e com início no número 10, devemos fazer o seguinte:

Efeito na página	Código HTML
X. Item 1 IX. Item 2 VIII. Item 3	<code><ol type="I" reversed start="10"> Item 1 Item 2 Item 3 </code>

As **listas não ordenadas** utilizam símbolos ao lado de cada um dos itens. Seu início deve ser marcado pela tag `` e seu término pela tag ``. UL é a abreviação de Unordered List (lista não ordenada). As *listas não ordenadas não possuem atributos válidos em HTML5*.

As listas ordenadas e não ordenadas podem ser aninhadas, de modo a formar sub-listas. Acompanhe o exemplo:

Efeito na página	Código HTML
<ul style="list-style-type: none"> Roupas <ol style="list-style-type: none"> Blusa azul Blusa verde Calça jeans Casaco Comida <ul style="list-style-type: none"> Pizza Refrigerante <ul style="list-style-type: none"> Coca-cola Guaraná Livros 	<pre> Roupas Blusa azul Blusa verde Calça jeans Casaco Comida Pizza Refrigerante Coca-cola Guaraná Livros </pre>

Tabelas

As **tabelas** também são elementos frequentemente utilizados em páginas web. Em HTML, as tabelas são criadas linha por linha.

A tag **TABLE** marca o início e o final da tabela.

A tag **TR** (= Table Row) marca o início e o final de cada linha da tabela.

A tag **TH** (= Table Header) marca os títulos de cada coluna da tabela. Ela também deve ser aberta e fechada a cada título.

A tag **TD** (= Table Data) marca o início e o final de cada campo (célula) da tabela.

Observe o código abaixo, que corresponde à criação de uma tabela com 3 linhas e 3 colunas:

Efeito na página	Código HTML									
<table><tr><th>Nome</th><th>Idade</th><th>Altura</th></tr><tr><td>Bruna</td><td>16</td><td>1,65m</td></tr><tr><td>Carlos</td><td>15</td><td>1,7m</td></tr></table>	Nome	Idade	Altura	Bruna	16	1,65m	Carlos	15	1,7m	<pre><table> <tr><th> Nome </th> <th> Idade </th> <th> Altura </th></tr> <tr> <td> Bruna </td> <td> 16 </td> <td> 1,65m </td></tr> <tr><td> Carlos </td> <td> 15 </td> <td> 1,7m </td></tr> </table></pre>
Nome	Idade	Altura								
Bruna	16	1,65m								
Carlos	15	1,7m								

Âncoras

As **âncoras (links)** são os pontos que ligam diferentes páginas entre si, com a finalidade de navegação. Por exemplo, ao clicar em um desses links, você é levado de uma página web para outra.

Eles ficam entre as tags **<a>** e ****, e possuem o atributo obrigatório **HREF**, que deverá conter o endereço da página de destino.

Outro atributo importante dos links é o **TARGET**, que especifica se aquele link, ao ser clicado, deve abrir na mesma janela ou em uma janela (aba) nova. O atributo TARGET pode assumir 4 valores diferentes (**_blank**, **_parent**, **_self** e **_top**), mas para o nosso curso, só utilizaremos os valores **_blank** (link abre em uma janela/aba nova) e **_self** (link abre na mesma janela). O valor **_self** é o padrão do HTML, ou seja, se você não especificar o TARGET, ele vai abrir o link na mesma aba em que você estava navegando.

Exemplo: quero inserir um link para a página www.cp2.g12.br, que abra em uma janela nova ao clicar no texto "site do Colégio Pedro II".

```
<a href="http://www.cp2.g12.br" target="_blank"> site do Colégio Pedro
II </a>
```

É importante colocar o <http://> na frente dos endereços de site! Teste o seguinte código de link e veja o que acontece:

```
<a href="www.cp2.g12.br" target="_blank"> site do Colégio Pedro II
</a>
```

Links

Os links não-âncora são responsáveis por estabelecer ligações entre documentos. Por exemplo, para ligar um arquivo HTML com arquivos CSS correspondentes.

```

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

<link rel="alternate stylesheet" href="contraste.css" title="Alto
Contraste">

```


Imagens

Para inserir imagens nas páginas, é necessário utilizar a tag ``. Assim como a tag de quebra de linha, a tag de imagem não fecha.

A tag `` possui dois atributos obrigatórios: o **SRC**, que contém o endereço da imagem a ser exibida, e o **ALT**, que deve oferecer uma descrição da imagem. Outros atributos importantes são **WIDTH** (largura) e **HEIGHT** (altura), para redimensionar a imagem.

Observe um exemplo para a inserção de uma imagem na página:

```

```

É possível usar uma imagem como link, desde que a tag `` fique entre as tags `<a>` e ``.

Referências

Ao criar páginas web, precisamos referenciar imagens, páginas e arquivos com grande frequência. Seja para inserir uma imagem, linkar uma página ou chamar um arquivo contendo CSS ou algum script.

Existem dois tipos de referências que podem ser feitas:

- **Referências absolutas**, quando linkamos para um arquivo ou site externo.
Exemplo: `http://google.com.br`
Sempre inicie a referência absoluta com `http://`
- **Referências relativas**, quando linkamos para um arquivo ou página em relação ao local em que estamos.
Exemplo: a página `pagina2.html` está na mesma pasta que `pagina1.html`. Para linkar de uma para a outra, não preciso dar o endereço completo dos arquivos.

```
<a href="pagina2.html" target="_self"> Página 2 </a>
```

Caso a página ou arquivo esteja em uma pasta interna, ainda é possível fazer a referência relativa.

```

```

Para voltar um nível de pasta, basta utilizar `../`

```
<a href="../textos.html" target="_blank"> Textos </a>
```

Formulários

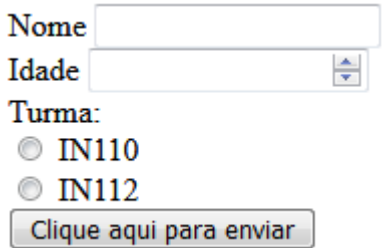
Formulários são estruturas utilizadas para a coleta de dados. Os dados inseridos em um formulário podem ser enviados para outras páginas, processados, salvos em bancos de dados, entre outras coisas.

Em HTML, formulário é tudo que fica entre as tags **<form>** e **</form>**.

Alguns dos principais atributos dentro da tag **<form>** são **method** e **action**. O atributo **method** pode assumir dois valores: GET ou POST, e significa o meio pelo qual os valores digitados nos formulários são enviados.

- GET é o padrão. Na passagem de dados, o que foi digitado fica visível no texto da url no par name-value. Nunca use GET para enviar senhas. Além disso, a capacidade de transmissão é limitada pelo tamanho da url, que aceita até 3000 caracteres.
- POST envia o dado por uma transação post do protocolo HTTP, de modo que os dados não ficam visíveis na url. Mais seguro e de tamanho ilimitado.

O atributo **action** representa a página para quais os dados serão enviados ou que contém os scripts que os processarão. No exemplo, o usuário será redirecionado para a página sucesso.html

Efeito na página	Código HTML
	<pre><form method="GET" action="sucesso.html"> <label>Nome<input type="text" name="nome"></label>
 <label>Idade<input type="number" name="idade"></label>
 Turma:
 <label><input type="radio" name="turma" value="IN110"> IN110</label>
 <label><input type="radio" name="turma" value="IN112"> IN112</label>
 <input type="submit" value="Clique aqui para enviar"> </form></pre>

Veremos como exibir e processar esses dados ao estudar JavaScript. Por enquanto, os formulários apenas exibirão uma mensagem de sucesso na página seguinte, sem funcionar de fato.

Tipos de campo

Como o exemplo mostra, a tag **input** é a responsável por exibir os principais elementos do formulário.

Os tipos de campos são diferenciados pelo atributo **type**, dentro da tag. Input é o campo genérico do formulário, que sempre seguirá o modelo

<input type="_____" name="_____">

O atributo **type** pode assumir diversos valores, os mais comuns são:

- Text: campo de texto simples
- Date: campo de seleção de data
- Number: campo de seleção numérica
- Password: campo de senha (tudo que for digitado vira bolinha)
- Email: campo de texto com validador de e-mail
- URL: campo de texto com validador de url (endereço de site)
- Checkbox: campo que permite que uma ou mais opções sejam marcadas por pergunta
- Radio: campo que permite que apenas uma opção seja marcada por pergunta

Campos do tipo checkbox e radio precisam obedecer às seguintes regras:

- Em uma mesma pergunta: possuem atributo name igual
- Em perguntas diferentes: possuem atributo name diferente

Além disso, tanto o checkbox quanto o radio são campos com valores pré-estabelecidos. Portanto, é importante colocar no input o atributo VALUE, que é o valor atribuído àquele item na resposta.

```
<input type="radio" name="questao1" value="a">
```

```
<input type="radio" name="questao1" value="b">
```

Os names podem ter qualquer nome. Preferencialmente, escolha nomes que façam sentido de acordo com o contexto. A mesma coisa com os values. Em campos do tipo checkbox, o name deve conter [] depois do nome escolhido.

```
<input type="checkbox" name="extra[]" value="extra1">
```

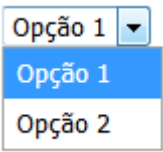
```
<input type="checkbox" name="extra[]" value="extra2">
```

Outros tipos de campo

Textarea é um campo de texto longo. Para criá-lo, basta usar

```
<textarea name="_____"></textarea>
```

O **select** é um campo que também envolve a escolha de apenas uma opção entre aquelas já previamente estabelecidas. Cada opção fica dentro de um **<option>**

Efeito na página	Código HTML
	<pre><select name="teste"> <option value="Opção 1">Opção 1</option> <option value="Opção 2">Opção 2</option> </select></pre>

Botões

Responsáveis por ações como enviar dados ou limpar todos os campos

Botão genérico:

```
<input type="button" value="">
```

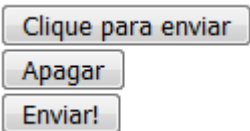
Botão de enviar:

```
<input type="submit" value="">
```

Botão de limpar todos os campos:

```
<input type="reset" value="">
```

Nos botões, o atributo VALUE altera o que você vê escrito no botão.

Efeito na página	Código HTML
	<pre><input type="submit" value="Clique para enviar">
 <input type="reset" value="Apagar">
 <input type="submit" value="Enviar!"></pre>

Não se esqueça de envolver cada campo com uma tag `<label></label>`. O texto externo do campo ficará atrelado a ele. Isso é importante principalmente por causa da acessibilidade, mas não produz efeitos visuais.

```
<label>Nome: <input type="text" name="seunome"></label>
```

CSS

Agora que aprendemos as principais tags HTML, vamos aprender a controlar o aspecto visual das nossas páginas web! Embora seja possível "embutir" esses atributos de estilo nas tags HTML, esta não é a prática recomendada pela W3C, além de não ser uma solução prática caso seja necessário alterar algum desses atributos depois.

Vamos criar arquivos CSS separados das nossas páginas, inserindo o `<link href="arquivo.css" rel="stylesheet">` dentro da tag **HEAD**, pois ela vai controlar a aparência dos elementos do **BODY**, mas o que está escrito dentro delas não será diretamente exibido na página web, e sim interpretado pelo navegador para que ele possa renderizar nossa página da forma que gostaríamos.

O HTML não deve conter tags para formatar textos, mas sim para definir o conteúdo de um documento. Por exemplo:

```
<h1>Cabeçalho da página</h1>
<p>Texto de um parágrafo. Este é um parágrafo. Quero pular uma
linha. <br> Pulei uma linha. </p>
```

O W3C não recomenda o uso da tag ``, e mesmo que os navegadores ainda interpretem as informações nela contidas, o uso de **FONT** já caiu em desuso. A forma correta de fazer toda a formatação de páginas web é utilizando CSS, utilizando um arquivo CSS externo.

Uma regra em CSS consiste na combinação de um seletor com um bloco de declarações. Por exemplo: quero que o plano de fundo da minha página seja o arquivo foto.jpg. A regra em CSS será:

```
body {background-image: url("foto.jpg");}
```

Neste exemplo, body é o nome do seletor, e o que está entre chaves é uma declaração. Caso eu também queira que o arquivo foto.jpg cubra toda a página, devo reescrever a regra da seguinte forma:

```
body {background-image: url("foto.jpg"); background-size: cover;}
```

Observe que agora o bloco entre chaves possui duas declarações. Não se esqueça de colocar o ponto-vírgula (;) ao final de cada uma das declarações. O CSS também não faz diferenciação de quebra de linha, então é possível pular linhas para deixar o código mais organizado:

```
body {
    background-image: url("foto.jpg");
    background-size: cover;
}
```

É possível incluir informações de estilo em uma página utilizando um arquivo .css externo. Para isso, é necessário inserir o seguinte código no arquivo HTML (também entre as tags `<head>` e `</head>`):

```
<link href="style.css" rel="stylesheet"/>
```

Sendo style.css o nome do arquivo com a extensão CSS.

Seletores

Seletores CSS são utilizados para referenciar os elementos HTML que serão modificados, com base no id, classe, atributo, nome do elemento, ou mais.

Seletores de elemento são baseados nos nomes dos elementos, isto é, nas tags de HTML. Portanto, todas as tags HTML que aprendemos até agora podem ser utilizadas como seletores: BODY, P, DIV, H1, TABLE, TD, TH, TR, IMG, A, FORM, INPUT, SELECT, TEXTAREA etc. Chamadas CSS nos seletores de elemento afetam todos os elementos marcados com a mesma tag.

Atenção:As tags HTML, HEAD e TITLE não devem ser utilizadas como seletores em CSS.

O exemplo abaixo mostra como alterar o estilo de todos os parágrafos de uma página, centralizando o texto e colocando-o na cor vermelha.

```
p {  
  text-align: center;  
  color: red;  
}
```

Seletores de ID utilizam o atributo ID de uma tag para referenciar elementos específicos. O ID de um elemento deve ser único dentro da página! Portanto, esse seletor é utilizado para apenas um elemento. Por exemplo, para alterar a aparência de apenas um parágrafo, é necessário dar um ID a ele e referenciar esse ID antecedido pelo símbolo # no arquivo .css.

O exemplo abaixo altera o estilo do parágrafo com ID "p1".

```
#p1 {  
  text-align: justify;  
  color: blue;  
}
```

Seletores de classe utilizam o atributo de classe para referenciar um conjunto de elementos específicos. A diferença entre classe e ID: enquanto o ID é único, a classe afeta um grupo de elementos. Portanto, o atributo classe (**class**) pode se repetir na mesma página. Para alterar a aparência de um conjunto de parágrafos, é necessário atribuir uma classe (class) a cada um deles e referenciar a **class** antecedido pelo símbolo . no arquivo .css.

O exemplo abaixo altera o estilo de todos os elementos com o atributo CLASS igual a "info".

```
.info {  
  text-align: center;  
  color: red;  
}
```

Também é possível especificar quais elementos serão afetados por uma declaração de classe. Se eu quiser que apenas as DIV com a classe "info" sejam afetadas, a declaração deve ficar assim:

```
div.info {  
    text-align: center;  
    color: black;  
}
```

Se você possuir mais de um elemento com a mesma definição de estilo, é possível agrupar as declarações. Por exemplo, se todos os cabeçalhos H1, H2 e H3 possuem letra vermelha, a declaração CSS pode ser feita assim:

```
h1, h2, h3 {  
    color: red;  
}
```

Comentários

Comentários em CSS são feitos entre `/*` e `*/`

```
h1, h2, h3 {  
    color: red; /*cor do h1, h2 e h3 é vermelha*/  
}
```

Cores

Cores podem ser especificadas de algumas maneiras diferentes:

- Utilizando **nomes pré-definidos** (red, orange, pink, green, tomato etc). Há 140 nomes pré-definidos¹.
- Valores em **hexadecimal**: sequência de 6 dígitos de 0 a F, sempre antecidos por #. Exemplo: #000000 corresponde ao preto e #FF0000, ao vermelho.
- **HSL** (hue, saturation, lightness): combinação de matiz, saturação e luminosidade. Por exemplo, `hsl(0,100%,50%)` corresponde ao vermelho.
 - Matiz/Hue varia entre 0 e 360 (0 = vermelho; 120 = verde; 240 = azul)
 - Saturação é uma porcentagem, que varia de 0% (tom de cinza) a 100% (cor)

¹ Lista disponível em: https://www.w3schools.com/colors/colors_names.asp

- Luminosidade também é uma porcentagem, com 0% sendo preto e 100%, branco.
- HSLA: combinação de matiz, saturação, luminosidade e alfa (transparência). O alfa varia de 0 (totalmente transparente) a 1 (totalmente opaco). Números decimais correspondem a cores translúcidas. Por exemplo, `hsl(0,100%,50%,0.5)` corresponde ao vermelho 50% opaco.
- RGB: combinação de vermelho, verde e azul. As cores variam de 0 a 255. Por exemplo, vermelho é `rgb(255,0,0)`
- RGBA: combinação de vermelho, verde, azul e alfa (transparência). As cores variam de 0 a 255, o alfa varia de 0 (totalmente transparente) a 1 (totalmente opaco). Números decimais correspondem a cores translúcidas. Vermelho com 50% de opacidade corresponde a `rgba(255,0,0,0.5)`

É possível modificar a cor de fundo de qualquer elemento com a declaração **background-color**

```
h1 {  
  background-color: #FF3399;  
}
```

É possível modificar a cor do texto de qualquer elemento com a declaração **color**

```
h2 {  
  color: Tomato;  
}
```

É possível modificar a aparência da borda de qualquer elemento com a declaração **border**, desde que na seguinte ordem: espessura (em pixels), estilo (solid - normal, dashed - tracejada e dotted - pontilhada) e cor.

```
p{  
  border: 2px solid #FF3399;  
}
```

Planos de fundo

Planos de fundo podem ser cores, como vimos acima, mas também podem conter imagens. Para que isso ocorra, é necessário utilizar a declaração **background-image**.


```
body{
  background-image: url("imgFundo.jpg");
}
```

As imagens de fundo, por padrão, se repetem tanto horizontalmente quanto verticalmente. Isso pode ser controlado pela declaração **background-repeat**.

```
body{
  background-image: url("imgFundo.jpg");
  background-repeat: repeat-x;
}
```

O exemplo acima irá repetir a imagem de fundo apenas horizontalmente. Para repetir apenas verticalmente, utiliza-se **repeat-y**. Para não repetir, utilizamos **no-repeat**.

Também é possível demarcar a posição que a imagem de fundo ficará na página, tanto horizontalmente quanto verticalmente, com **background-position**.

```
body{
  background-image: url("imgFundo.jpg");
  background-repeat: repeat-x;
  background-position: center top;
}
```

No exemplo acima, a imagem estará centralizada horizontalmente, mas no topo da página. Horizontalmente, ela poderá se posicionar à direita (**right**), esquerda (**left**) ou centro (**center**). Verticalmente, ela poderá se posicionar acima (**top**), abaixo (**bottom**) ou no centro (**center**).

Para que a imagem de fundo não role conforme a rolagem da página, utilizamos a declaração **background-attachment** com o valor **fixed**.

```
body{
  background-image: url("imgFundo.jpg");
  background-repeat: repeat-x;
  background-position: center top;
  background-attachment: fixed;
}
```

Margens

Margens são espaços criados ao redor dos elementos, isto é, fora das bordas definidas. Margens podem ser definidas com valores absolutos (em pixels, cm, pt ou outras unidades), valores percentuais, ou até automaticamente (auto), isto é, calculadas pelo navegador.

É possível atribuir valores negativos às margens.

```
div{
  margin-top:15px;
  margin-bottom: 20px;
  margin-right: 30px;
  margin-left: 30px;
}
```

Padding

Padding (enchimento) gera espaço ao redor dos conteúdos de um elemento, isto é, dentro das bordas definidas. Esses valores também podem ser especificados em qualquer unidade. Ao contrário das margens, o padding não aceita valores negativos.

```
p{
  padding-top:10px;
  padding-bottom: 15px;
  padding-right: 100px;
  padding-left: 100px;
}
```

Tamanho

O tamanho dos elementos é controlado pela altura (**height**) e largura (**width**). Esses valores também podem ser especificados em qualquer unidade.

```
img.foto{
  height: 100px;
  width: 100px;
}
```

Além disso, existem propriedades como **max-height** e **max-width**, que estipulam altura e largura máximas para um elemento. Isso é muito importante para criar páginas que se ajustam a telas de tamanhos diferentes. Da mesma forma, existem **min-height** e **min-width**, que estipulam altura e largura mínimas.

Formatação de texto

Além da cor, é possível formatar textos de outras formas. Por exemplo, modificando o alinhamento do texto com **text-align** ou modificando a decoração do texto com **text-decoration**. Normalmente, o text-decoration é utilizado para retirar o sublinhado de links em alguns sites.

```
p{
  text-align: left;
  text-decoration: none;
}
```

O texto pode ser alinhado à esquerda (left), direita (right), ao centro (center) ou justificado (justify).

Para modificar a fonte de um texto, utiliza-se o **font-family**. A propriedade **font-style** pode deixar o texto normal ou itálico (italic). Para controlar o tamanho da fonte, utiliza-se **font-size**. A propriedade **font-weight** controla o peso da fonte (bold = negrito).

```
p{
  font-family: "Times New Roman", serif;
  font-style: italic;
  font-size: 15px;
  font-weight: bold;
}
```

Links

É possível alterar o CSS de links com todas as propriedades já mencionadas anteriormente: cor, fundo, fonte etc.

Além disso, é possível controlar a aparência dos links de acordo com os estados deles. São 4 estados possíveis, que devem aparecer nessa exata ordem:

- **a:link** - link normal que ainda não foi visitado pelo usuário
- **a:visited** - link que foi visitado pelo usuário
- **a:hover** - quando o mouse passa por cima do link
- **a:active** - link no momento em que é clicado

```
a:link{
  font-weight: bold;
}

a:visited{
  font-weight: bold;
  color: #FF3399;
}

a:hover{
  font-weight: bold;
  text-decoration: overline;
}

a:active{
  color: tomato;
  background-color: white;
}
```

Posicionar elementos

O atributo **float** é utilizado para posicionar elementos na página. Ele pode assumir três valores: **left** (esquerda), **right** (direita) ou **none** (sem efeito). É possível utilizá-lo para definir a posição de DIVs na página, assim como para outros propósitos.

Para centralizar elementos, podemos utilizar **margin: auto** (só tem efeito visualmente se o elemento possuir uma declaração **width** com valor inferior a 100%).

Box Model

Todos os elementos HTML podem ser considerados como retângulos (ou caixas). O termo "Box Model" é utilizado quando falamos de projetar layouts e se refere ao espaço que um elemento verdadeiramente ocupa.

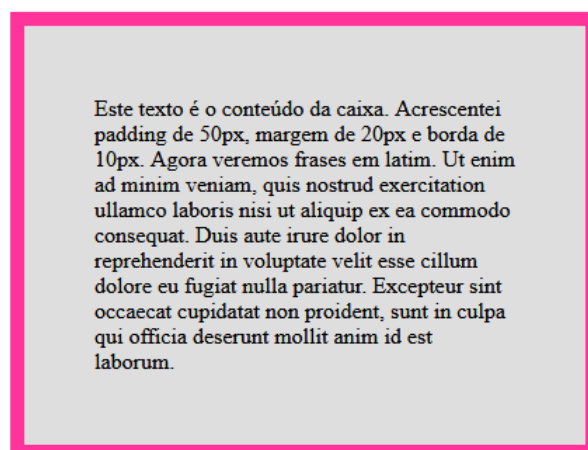
Por exemplo, criar um div com 300px de largura significa que todo o conteúdo dentro daquela div terá 300px de largura. Ao acrescentar propriedades como Padding (enchimento), Border (borda) e Margin (margem) nessa mesma div, suas dimensões se modificarão. Padding e margens sempre são transparentes.

No exemplo abaixo, o parágrafo possui borda de 1px e 0 de margem. O elemento abaixo dele é uma div com 20px de margem (observe que ela "começa" um pouco depois do parágrafo), 10px de borda rosa e 20px de enchimento/padding. A largura da div é de 300px, mas na realidade ela ocupa 460px. Pode ser feito o mesmo cálculo para a altura, que não foi especificada na div, mas independente disso, terá mais 160px do que o espaço do conteúdo.

Margem + borda + padding (esquerda)	Elemento	Margem + borda + padding (direita)	Total
20 + 10 + 50 = 80px	300px	20 + 10 + 50 = 80px	80 + 300 + 80 = 460px

Box Model

Todos os elementos HTML podem ser considerados como retângulos (ou caixas). O termo "Box Model" é utilizado quando falamos de projetar layouts e se refere ao espaço que um elemento verdadeiramente ocupa.



ACENTUAÇÃO E CARACTERES ESPECIAIS

Para evitar a ocorrência de caracteres indesejados em suas páginas, escreva os caracteres que são acentuados de acordo com o especificado na tabela abaixo.

A estrutura dos caracteres especiais é a seguinte:

& [caracter] [acento] ;

Por exemplo, para escrever o ç, é necessário fazer ç - > c é o caracter e cedil é o nome do cedilha em inglês.

Os principais "acentos" são: circ (circunflexo), grave(acento grave - o que indica a crase), acute (agudo), uml (trema) e tilde (til). Segue tabela com os caracteres especiais mais comuns:

Á Á	Í Í	Ú Ú
á á	í í	ú ú
Â Â	Î Î	Û Û
â â	î î	û û
À À	Ì Ì	Ù Ù
à à	ì ì	ù ù
Å Å	Ĩ Ï	Ü Ü
å å	ĩ ï	ü ü
Ã Ã	Ó Ó	< <
ã ã	ó ó	> >
Ä Ä	Ô Ô	& &
ä ä	ô ô	" "
Æ Æ	Ò Ò	® ®
æ æ	ò ò	© ©
É É	Ø Ø	Ý Ý
é é	ø ø	ý ý
Ê Ê	Õ Õ	Þ Þ
ê ê	õ õ	þ þ
È È	Ö Ö	ß ß
è è	ö ö	
Ë Ë	Ç Ç	
ë ë	ç ç	
Ð Ð	Ñ Ñ	
ð ð	ñ ñ	