

GLOSSÁRIO

HTML & CSS

Saiba como construir websites



Consultas rápidas

Utilize este documento para consultar termos utilizados de forma recorrente no curso e facilitar o seu aprendizado.

Lembre-se de conversar com suas colegas, vizinhas e familiares sobre o que está aprendendo no Potência Feminina.

Colabore

Ouviu algum termo no curso e gostaria de vê-lo aqui no glossário? Converse a pessoa que está facilitando este curso em sua ONG ou envie uma mensagem para equipe do Potência Feminina.

Agradecemos a sua ajuda



HTML: sigla para Hyper Text Markup Language, que significa linguagem de marcação de hipertexto. Ela é utilizada para desenvolver páginas e documentos eletrônicos para a internet, garantindo a formatação ideal para sites.

CSS: sigla para Cascading Style Sheets, é uma "folha de estilo" composta por "camadas" e utilizada para definir a apresentação (aparência) em páginas da internet que adotam para o seu desenvolvimento linguagens de marcação (HTML).

Framework: é um conjunto de códigos prontos com diversas funções que podem ser utilizadas no desenvolvimento de sites. Sobretudo, o objetivo de uso dessa ferramenta é aplicar funcionalidades e estruturas já programadas para garantir mais produtividade e qualidade no desenvolvimento de um projeto.



HTML

As marcações que são utilizadas para estruturar o código chamam-se tags. Elas servem para indicar o que será mostrado. Por exemplo: `<head></head>` indica que nesta área será inclusa o conteúdo relacionado ao cabeçalho do site. `<body></body>` indica que nesta área será inclusa o conteúdo do site. `<h1></h1>` indica o título de um post de um blog, por exemplo.

```
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>Curso RME e WoMakersCode</title>
5    </head>
6    <body>
7      <h1>Título do site</h1>
8      <p>Descrição do site</p>
9    </body>
10 </html>
```

Estrutura Básica – HTML

- **DOCTYPE** – é o que avisa dos browsers e navegadores de busca que tipo de documento está para carregar.
- **HTML** – tudo o que estiver com as tags <html> é porque o código será escrito em html.
- **HEAD** – aqui nós indicamos o título do site, as meta tags (SEO), se o site será indexado nos sites de busca ou não, entre outros.
- **TITLE** – dentro da tag <head> nós incluimos o <title>, que é onde definimos o nome do site.

BODY – dentro dela é que vamos incluir todo o nosso código em html do site.

```
1  <!DOCTYPE html>
2  <html>
3  <head>
4    <title>Curso RME e WoMakersCode</title>
5  </head>
6  <body><!-- conteudo do site --></body>
7  </html>
```

Tags

O que são: é uma marcação onde com ela podemos definir o que colocar na página (seja um título `<h1>` ou um parágrafo `<p>` dentro do corpo da página `<body>`, por exemplo).

Com o HTML5 temos novas tags estruturais para definir o cabeçalho e rodapé do nosso site. São eles: `<header>` e `<footer>`. Há outros também, como o `<nav></nav>`, que são tags de navegação, muito utilizado para estruturar menus.

Metadados: As tags de metadados são necessárias para organizar as informações do conteúdo de forma mais prática, inteligível não só para o seu site ou blog, mas também pelos sites de busca.

- **META CHARSET** – é o que define o tipo de codificação de caractere a ser utilizado, podendo ser UTF-8 (o mais utilizado) ou ISO-8869-1.
- **META DESCRIPTION** – a descrição do seu site.
- **META KEYWORDS** – as palavras-chave do site que irão aparecer nos sites de busca.
- **META AUTHOR** – o autor do site.
- **META VIEWPORT** – é o que define quando vamos trabalhar em um site responsivo.

```
<meta charset="UTF-8">  
<meta name="description" content="Meu primeiro site">  
<meta name="keywords" content="HTML5, CSS3">  
<meta name="author" content="WoMakersCode">
```

Estruturais: são necessárias para a estruturação do nosso site. Além dessas há também uma que é necessária para o desenvolvimento em camadas: a tag <div>.

No exemplo usamos a tag <div> para definir o bloco de conteúdo.

Aqui podemos atribuir uma classe ou id para formatar a nossa área.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Meu primeiro site
    </title>
  </head>
  <body>
    <header>
      <nav>
        <ul>
          <li>Item 1
          </li>
        </ul>
      </nav>
    </header>
    <div>
      
      <h1>Título
      </h1>
      <p>Descrição
      </p>
    </div>
    <footer>
      <p>2021. WomakersCode
      </p>
    </footer>
  </body>
</html>
```


Texto: como o próprio nome já diz, são utilizadas para indicar que aquele item pode ser um título, subtítulo, descrição, texto corrido, entre outros. Geralmente usamos os h's para definir títulos e subtítulos, por exemplo. Vai do <h1> até o <h6>. O <p> usamos para definir um bloco comum de texto parágrafo e o para observações, textos de pouca importância. A tag <small> serve para marcar textos menores. A tag (strong é mais utilizado) serve para deixar o texto em negrito (bold).

```
<h1>Título</h1>  
<h4>Subtítulo</h4>  
<p><strong>Lorem ipsum</strong> lorem lorem lorem</p>
```



- A tag `` também serve para indicar um texto em negrito*
- Já a tag `<u>` serve para indicar um texto sublinhado*
- A tag `<i>` serve para indicar um texto em itálico*

Atualmente as tags ``, `<u>` e `<i>` não são muito utilizadas para definir um atributo para o texto. Neste caso, usamos o CSS. O caso da tag `<i>` é que hoje em dia ela é mais usada para indicar um ícone.

Imagens, vídeos e conteúdos: as tags mais conhecidas para incorporar algum conteúdo é o e <iframe>. O primeiro permite a inclusão do caminho de uma imagem e inserir alguns atributos nela.

```

```

Repare que dentro da tag usamos os atributos width e height para indicar a largura e altura, respectivamente. O alt e o title servem para indicar do que a imagem se trata. Usa-se bastante para facilitar a acessibilidade.

Usamos a tag <iframe> para incorporar algum vídeo ou conteúdo externo que a gente queira exibir em alguma página.

```
<iframe width="560" height="315" src="https://www.youtube.com/embed/WwoKkq685Hk" frameborder="0" allowfullscreen></iframe>
```

No exemplo acima, usamos o embed de vídeo do Youtube. Perceba que aqui também usamos os atributos de largura e altura e também o frameborder (que indica se você quer que apareça uma borda no frame do vídeo – neste caso, colocamos 0) e o allowfullscreen (adicionando esse atributo dentro da tag ele permite que você possa visualizar o vídeo em tela cheia). Podemos também usar a tag <iframe> para incluir algum conteúdo externo dentro da nossa página.

```
<iframe src="http://www.google.com.br" width="1024" height="768"></iframe>
```

Formulário: A tag <form> indica que dentro dela se encontra a estrutura de um formulário.

```
<form>
  <label>Nome</label>
  <input type="text" value="nome" id="" />
  <label>Email</label>
  <input type="text" value="email" id="" />
  <label>Telefone</label>
  <input type="number" value="tel" id="" />
  <label>Selecione o seu estado</label>
  <select name="estado">
    <option value="rj">Rio de Janeiro</option>
    <option value="sp">São Paulo</option>
    <option value="mg">Minas Gerais</option>
    <option value="es">Espírito Santo</option>
  </select>

  <label>Mensagem</label>
  <textarea name="message" rows="10" cols="30"></textarea>
  <button type="submit">Enviar </button>
</form>
```



- **INPUT** – um dos elementos mais importantes. Dentro dele podemos indicar alguns atributos usando o `type=""`. Logo abaixo estão os mais usados:
 - `type="text"` – como o próprio nome já diz, indica que dentro daquele input é permitido caracteres alfanuméricos
 - `type="email"` – indica que só será aceito emails
 - `type="number"` – indica que dentro do input só poderá incluir números
 - `type="checkbox"` – indica que este input terá a funcionalidade de um checkbox
 - `type="radio"` – indica que este input terá a funcionalidade de um radio button
 - `type="submit"` – indica que este input terá a funcionalidade de envio (usa-se a tag `<button>`)
 - `type="password"` – indicado para senhas
 - `type="file"` – indica que dentro desse input dará a função do usuário incluir algum arquivo.
- **SELECT** – elemento que serve para indicar que dentro dele haverá um box listando opções. E dentro desse `<select>` usamos a tag `<option>`
- **TEXTAREA** – elemento que indica campo de texto.

Tabela: Usamos a tag <table> para indicar que dentro dela haverá um conteúdo de numa tabela.

```
<label>Mensagem</label>
<textarea name="message" rows="10" cols="30"></textarea>
<button type="submit">Enviar </button>
</form>

<table>
  <tr>
    <th>Nome</th>
    <th>Sobrenome</th>
    <th>Idade</th>
  </tr>
  <tr>
    <td>Aline</td>
    <td>Bezzoco</td>
    <td>26</td>
  </tr>
  <tr>
    <td>Eduarda</td>
    <td>Gomes</td>
    <td>32</td>
  </tr>
</table>
```

TR – define uma linha da tabela TH – define o cabeçalho da tabela TD – define uma célula da tabela

Style e Script: Usamos as tags style e script para indicar que dentro dela haverá uma folha de estilo (CSS) e o outro um script em jQuery ou Javascript, respectivamente.

```
<style type="text/css">
|   html, body {
|       background-color: ■ white;
|       overflow: hidden;
|   }
</style>
<script>
|   $(('.minha-classe')).find(('a')).addClass(('.estilo-link'));
</script>
```

Link: Para indicarmos que aquele texto ou palavra será um link usamos a tag <a>.

```
<a href="http://www.google.com.br" target="_blank"></a>
```

O atributo href="" serve para indicar qual é a url que você irá incluir. Neste caso usamos o site do Google.

Perceba também que dentro da tag <a> usei o atributo target=". Ele indica onde a página irá abrir. Neste caso, pedi para que ela abrisse em uma nova página (_blank). Mas você também pode pedir para que ela carregue na mesma página. Para isso usa-se o _parent. Assim:

```
<a href="http://www.google.com.br" target="_parent"></a>
```

Listas: Para criarmos uma lista em HTML usamos as tags e . Usamos a para indicar o início da lista e o o conteúdo que estará dentro dela.

```
<ul>
  <li>Item 01</li>
  <li>Item 02</li>
  <li>Item 03</li>
</ul>
```

Atributos: antes de surgir o CSS, muito se utilizava os atributos dentro dessas tags para definir uma formatação. Ele se refere às características de cada tag. Por exemplo: a tag Ela serve para definir a imagem a ser exibida na página. Dessa forma:

```

```

Mas se quisermos definir a largura e o tamanho dela devemos atribuir também o width="" e height="". O src="" também é um atributo, pois estamos definindo qual é o caminho desta imagem.

```

```

Tag <div>: vem do nome em inglês divide que significa dividir. Ou seja, a div serve para dividir a nossa estrutura em blocos, alterar o estilo ou comportamento dela em partes específicas da página, posicionar objetos e armazenar dentro dela diversos elementos como as tags HTML que já vimos nos capítulos anteriores, como , <a>, <h1> entre muitos outros. Na div podemos atribuir uma ID ou CLASS (que servem como identificação/marcação necessária para os ajustes que pretendemos fazer) que você verá logo abaixo a diferença entre as duas. ID – as id's são únicas. Cada elemento poderá conter apenas 1 id.

```
<div id="menu">
  <ul>
    <li>Item 01</li>
    <li>Item 02</li>
    <li>Item 03</li>
  </ul>
</div>
```

Não é recomendado usar várias id's dentro uma página ou bloco. Se você quiser atribuir um nome a outro bloco é necessário usar o class.CLASS – ao contrário das id's, as classes não são únicas e você pode atribuí-las a qualquer elemento. Você também pode usar a mesma classe para outros elementos. É necessário usá-la para definir os nossos estilos no CSS. Atribuindo um nome para ela podemos aplicar o CSS dentro dela e customizar o bloco selecionado.

```
<div class="conteudo">  
  <div class="destaques">  
      
  </div>  
</div>
```

Elementos HTML

- `<html>` : Representa a raiz de um documento HTML ou XHTML. Todos os outros elementos devem ser descendentes desse elemento.
- `<head>` : Representa uma coleção de metadados sobre o documento, incluindo links suas definições de scripts e folhas de estilo.
- `<title>` : Define o título do documento , apresentado na barra de título do navegador ou na guia da página. Ele só pode conter texto e todas as tags contidas não são interpretados.
- `<link>` : Usado para linkar JavaScript e CSS externo o documento HTML.
- `<meta>` : Define os metadados que não podem ser definidas usando outro elemento HTML.
- `<script>` : Define um script interno ou link para um script externo. A linguagem de script é JavaScript.
- `<body>` : Representa o principal conteúdo de um documento HTML. Há apenas um elemento em um documento.
- `<section>` : Define a seção do Documento.

- `<nav>` : Define uma seção que contém apenas links de navegação.
- `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>`, `<h6>` : São elementos que representam os seis níveis de títulos de cabeçalhos dos documentos. Um elemento título descreve brevemente o tema da seção.
- `<header>` : Define o cabeçalho de uma página ou seção. Muitas vezes contém um logotipo, o título do site e uma menu de navegação do conteúdo.
- `<footer>` : Define o rodapé de uma página ou seção.
- `<p>` : Define uma parte que deve ser exibido como um paragrafo.
- `<hr>` : Linha horizontal.
- `` : Define uma lista não ordenada.
- `<div>` : Representa um contêiner genérico sem nenhum significado especial.
- `<a>` : Representa um hyperlink, ligando a outro recurso.
- `` : Define um texto em negrito.
- `<small>` : Representa lado comentário , que é o texto como um aviso legal, um autor que não é essencial para a compreensão do documento.

- `` : Define um texto em negrito.
- `<i>` : Define um texto em itálico.
- `` : Define uma seção no documento.
- `
` : Quebra de linha.
- `` : Representa uma imagem.
- `<iframe>` : Define a inclusão de algum conteúdo externo para ser exibido dentro de uma página.
- `<table>` : Define uma tabela
- `<tbody>` : Define o corpo da tabela.
- `<td>` : Define uma célula da tabela.
- `<tr>` : Define uma linha da tabela.
- `<th>` : Define o cabeçalho da tabela.
- `<tfoot>` : Define o rodapé da tabela.
- `<thead>` : Define o cabeçalho da tabela.
- `<form>` : Define um formulário.

- `<label>` : Define uma "label" para o formulário.
- `<input>` : Define um campo de inserção.
- `<button>` : Define um botão de comando.
- `<select>` : Define uma lista selecionável.
- `<option>` : Define uma opção em uma lista suspensa (drop-down list).

CSS

O que é: CSS significa em inglês Cascading Style Sheets (Folhas de Estilo Encadeadas) e é através dele que conseguimos customizar o HTML, definir um estilo para ele.* Recapitulando: O HTML serve para estruturar a nossa página que vamos implementar, inserir o conteúdo.

Chamando o CSS em uma página HTML: Para chamar um arquivo CSS dentro da nossa página HTML devemos usar a tag `<link href="" rel="">`. Assim:

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

Neste caso é necessário criar um arquivo com a extensão .css. Há outras formas de incluir o CSS no HTML: interno, incluindo as tags `<style></style>` antes de fechar a tag `</head>` e inline, dentro do elemento. Exemplo: `<p style="font-size: 12px;"></p>`

Sintaxe e Seletores: O CSS trabalha no formato propriedade: valor a partir de uma tag ou classe definida no HTML.

```
a {  
  text-decoration: none;  
}
```

No exemplo acima atribuímos uma propriedade dentro do seletor "a" indicando um valor. Neste caso, o valor é o text-decoration:none, que permite que o link não apresente bordas ou cores diferentes além do que foi estilizado para ele.

Outra coisa que vale frisar é que sempre tem que iniciar e fechar com as chaves e usar o ponto e vírgula para cada propriedade:valor que definir. Assim:

```
a {  
  text-decoration: none;  
  cursor: pointer;  
}
```

Seletores: São os elementos que precisamos para indicar que tipo de propriedade e valor eles vão ter. Podemos dividi-lo da seguinte forma:

- Global: com ele selecionamos todas as tags de um mesmo tipo.

```
p {  
  color: #000000;  
}
```

- Classe: aplica as propriedades específicas para a classe indicada.

```
.destaque {  
  background-color: #000000;  
  width: 800px;  
  height: 600px;  
}
```

Cascata e Herança: O CSS trabalha em cascatas, então se dentro da nossa estrutura <div class="conteudo"> tiver um <h1>, <p> e até mesmo uma outra <div> dentro dela ambos irão herdar as propriedades definidas na classe "conteúdo".

HTML

```
<div class="conteudo">
  <h1>Titulo</h1>
  <p>Descrição</p>
  <div class="destaque"></div>
</div>
```

CSS

```
.conteudo {
  color: #ffffff;
  background-color: #ffff00;
  font-size: 14px;
}
```


Perceba que dentro do seletor da classe conteúdo foi atribuído propriedades e valores que acabam refletindo para tudo o que estiver dentro dela. Ou seja, essas propriedades foram herdadas pelos elementos internos (ou como alguns costumam chamar, de “pai para filho”). Mas você pode especificar também que o <p> dentro da div “conteúdo” terá um comportamento diferente, por exemplo.

```
.conteudo {  
  color: #ffffff;  
  background-color: #ffff00;  
  font-size: 14px;  
  width: 960px;  
  height: 720px;  
}  
  
.conteudo p {  
  color: #000;  
  font-size: 14px;  
}
```

Aqui estilizamos o parágrafo dentro da classe conteúdo. Perceba que a hierarquia se manteve usando o “pai”, a classe “conteúdo”, na frente do <p>, o seu “filho”. Assim indica que SOMENTE dentro dessa classe o <p> terá este comportamento. Se deixássemos ele fora significaria que TODOS os elementos <p> teriam cor branca e o tamanho da fonte 14px.

Pseudo-Classes: Existem diversas pseudo-classes. Usa-se muito para atribuir propriedades nos links como definir o estilo dele quando estiver ativo ou depois que foi clicado, o mouseover, entre outros. Há também aqueles que podemos definir o estilo apenas para o primeiro bloco ou o último, por exemplo. A pseudo-classe usa-se da seguinte forma:

```
a {  
  color: #ff0000;  
}  
a:hover {  
  color: #000000;  
  text-decoration: none;  
  font-weight: bold;  
}
```

Pseudo-classes mais utilizadas quando atribuímos algo para a tag <a>:

- **a:visited** – quando necessitamos estilizar quando o link já foi clicado (o default é ele ficar na cor roxa);
- **a:hover** – quando necessitamos estilizar o mouseover do nosso link (que é quando passamos o mouse em cima dele e tem um efeito diferente – o default é aparecer sublinhado);
- **a:focus** – quando necessitamos estilizar o “foco” dele quando usamos o tab para destacá-lo, por exemplo (costuma ficar uma borda azul);
- **a:active** – quando necessitamos estilizar o link quando está sendo clicado ou quando o enter é pressionado. Um exemplo explica bem a função dele: quando precisamos deixar o item de um menu ativo de acordo com a página dele.

Propriedades CSS para aplicar estilos em um bloco de texto

- **Text-decoration:** propriedade que define o tipo de “decoração” que o texto irá ter. Poderá ter sublinhados, riscos no texto;
- **Text-transform:** propriedade que define a “transformação” do texto. Poderá ficar todo em caixa alta (letras maiúsculas) ou em caixa baixa (letras minúsculas), por exemplo.
- **Color:** propriedade que define a cor de um texto;
- **Font-size:** propriedade que define o tamanho da fonte do texto;
- **Font-weight:** propriedade que define o “peso” da fonte. Pode ser bold (negrito), regular (normal);
- **Font-style:** propriedade que define o estilo da fonte, podendo ser italic (itálico) ou normal.

Propriedades CSS para customizar um bloco de texto

- **font-family:** define a família da fonte a ser utilizada;
- **font-style:** propriedade que define o estilo da fonte, podendo ser italic (itálico), oblique ou normal.
- **font-weight:** define o “peso” da fonte. Pode ser bold (negrito), regular (normal) ou ser definido por peso em números dependendo da fonte. Geralmente vai de 100 a 900. Os mais usados são 300, 400, 500, 600, 700, 800 (seria do light até o extrabold)
- **font-size:** define o tamanho da fonte do texto (o mais usado é o tamanho em px, mas é aceito também em pt, em, %);
- **text-align:** define o alinhamento do texto, podendo ser alinhado à esquerda (left), centralizado (center), à direita (right) ou justificada (justify);
- **color:** propriedade que define a cor de um texto podendo ser em hexadecimal, RGB ou o nome da cor em inglês.

Exemplo:

```
.conteudo h1 {  
  color: red;  
  font-size: 20px;  
  text-align: center;  
}  
.conteudo p {  
  color: #000000;  
  font-size: 14px;  
  text-align: left;  
}
```

Margens e Bordas: Na hora de customizar o nosso bloco (seja de estrutura usando a div ou de conteúdo, que é dentro dele) podemos aplicar margens internas e externas, assim como as bordas. Para isso usamos as seguintes propriedades:

- **Margin:** para definir as margens externas, seja superior (top), inferior (bottom), à esquerda (left) ou à direita (right). Podemos defini-la separadamente ou todas juntas.
- **Border:** serve para estilizar o contorno do bloco. Seguindo os mesmos padrões do padding e margin, podemos defini-lo superior (top), inferior (bottom), à esquerda (left) ou à direita (right) ou todos eles juntos.

Exemplos Margin:

```
.conteudo .destaque {  
  width: 800px;  
  height: 600px;  
  margin-top: 10px;  
  margin-bottom: 10px;  
}
```

```
.conteudo .destaque {  
  width: 800px;  
  height: 600px;  
  margin: 10px 0;  
}
```



Os dois exemplos anteriores explicam como podemos declarar os valores da propriedade margin. Seja separadamente ou simplesmente usando margin. Ao usar apenas o margin, fique atento a seguinte ordem: TOP – RIGHT – BOTTOM – LEFT. Observe no segundo exemplo que o margin está 10px 0. Significa que apenas atribuí valores apenas para as margens superior e inferior enquanto a da esquerda e direita estão zeradas. Nestes dois exemplos também usei duas propriedades: width e height. Ambos servem para definir a largura e altura de um elemento. Podemos usar em px, em, %..Para definir as margens internas usando a propriedade padding. Assim o margin, podemos definí-la superior (top), inferior (bottom), à esquerda (left) ou à direita (right) ou todas elas juntas.

```
.conteudo .destaque p {  
  padding: 0 10px;  
}
```

```
.conteudo .destaque p {  
  padding-left: 10px;  
  padding-right: 10px;  
}
```

Da mesma forma que o margin nós podemos atribuir os valores separadamente usando as propriedades específicas ou definindo apenas por 1 propriedade seguindo a mesma ordem do top – right – bottom – left .

Exemplos Border:

```
.conteudo .destaque {  
  width: 800px;  
  height: 600px;  
  margin: 10px 0;  
  border: 1px solid #000;  
}
```

```
.conteudo .destaque {  
  width: 800px;  
  height: 600px;  
  margin: 10px 0;  
  border-bottom: 1px solid #000;  
}
```

Perceba que no primeiro exemplo foi usado a propriedade border e atribuindo os seguintes valores: largura (width) estilo da borda (style) e a cor (color). Já no segundo exemplo específico que apenas a borda será customizada na parte inferior (bottom).

Propriedade Background: define tudo o que é relacionado ao fundo de exibição. Ela possui algumas variações

Exemplo 1 – Background-color define o fundo através de uma cor sólida:

```
.conteudo {  
|   background-color: green;  
|  
}
```

Exemplo 2 – Background-image define o fundo através de uma imagem:

```
.conteudo {  
|   background-image: url('img/bg.jpg');  
|  
}
```

Exemplo 3 – Background-repeat define se a imagem de fundo se repetirá

```
.conteudo {  
  background-image: url('img/bg.jpg');  
  background-repeat: repeat;  
}
```

```
.conteudo {  
  background-image: url('img/bg.jpg');  
  background-repeat: no-repeat;  
}
```

Os dois exemplos acima usam a propriedade background-repeat, porém com valores diferentes. Enquanto o primeiro (repeat) permite que o fundo se repita, o outro não permite. Mas há outros valores também: repeat-x (apenas horizontal) e repeat-y (apenas vertical).

Exemplo 4 – Background-position define a posição do background. É muito utilizado também quando se prepara imagens em sprites (que é um arquivo png com diversas imagens) e aí é definido o posicionamento do mesmo para a imagem aparecer.

```
.conteudo {  
  background-image: url('img/bg.jpg');  
  background-position: center;  
}
```

```
.conteudo {  
  background-image: url('img/bg.jpg');  
  background-position: 10px -20px;  
}
```

Os dois exemplos acima explicam como podem ser aplicados a propriedade background-position. A primeira explica que podemos usar os valores, left, right e center para definir o posicionamento do fundo. Neste caso pedi para que a posição do background seja centralizado. Já o segundo o posicionamento é feito através de valores numéricos. O primeiro valor indica o posicionamento vertical e o segundo, horizontal.

Exemplo 5 – Define o tamanho do background.

```
.conteudo {  
  background-image: url('img/bg.jpg');  
  background-size: 100%;  
}
```

```
.conteudo {  
  background-image: url('img/bg.jpg');  
  background-size: 100% auto;  
}
```

Aqui pedimos para definir os valores do tamanho do background em porcentagem. Enquanto o primeiro é 100% (ou seja, largura e altura) o segundo definimos apenas 100% a largura e auto a altura.

Propriedade Display: A propriedade display uma das mais importantes para controlar o layout. Através dela podemos definir se vamos exibir o nosso conteúdo lado a lado ou por quebra de linha ou obter um formato tabela ou simplesmente fazer com que aquele bloco desapareça, por exemplo. Nesta apostila vamos citar os quatro valores mais usados: block, inline, inline-block e none. No Guia de Referência no final da apostila contém os valores completos dessa propriedade tão útil para o desenvolvimento de uma página.

- BLOCK: é o valor mais usado. Quando utilizado, indica que este é um elemento de nível de bloco e indica uma nova quebra de linha, iniciando logo abaixo de outro bloco independente da posição (esquerda, direita ou centralizado). Exemplo:

```
.conteudo .destaque {  
  display: block;  
  width: 800px;  
  height: 600px;  
}
```

- NONE – indica que quando atribuirmos esse valor informamos que não queremos que aquele bloco não apareça, ou seja, esconde o elemento sem precisar remover o código. Exemplo:

```
.conteudo .destaque {  
  display: none;  
  width: 800px;  
  height: 600px;  
}
```

- INLINE – usado para indicar um elemento de linha. Usa-se muito em tags como ou <a> para indicar que aquele conteúdo dentro de um parágrafo irá permanecer na mesma linha sem precisar realizar a quebra do mesmo. Exemplo:

```
header .menu ul li {  
  display: inline;  
  color: black;  
}
```

No exemplo acima aplicamos uma customização bem comum: transformando o menu no formato horizontal, já que por padrão, o li tem comportamento de lista e por si próprio já fica um embaixo do outro. Aqui mudamos isso. Atribuímos o valor inline para que cada li ficasse um do lado da outra.

Diferenças entre Visibility e Display: visibility e display dão a ideia de possuírem a mesma finalidade, porém são propriedades completamente diferentes. O visibility serve para indicar se um elemento será visível ou não dentro uma página (por exemplo), mas sem a sua estrutura. O `visibility:visible` indica para exibir e o `visibility:hidden` para desaparecer. Já o `display:block` indica uma quebra de bloco e o `display:none` faz com que o elemento desapareça da estrutura, alterando o seu comportamento, não ocupando o espaço em que ele estava visível ao contrário do `visibility:hidden`.

Propriedade Float: permite a retirada de um elemento que está seguindo uma estrutura normal no documento (no nosso caso, no HTML) fique flutuando seja à esquerda ou à direita do restante do conteúdo e fazendo com que os outros elementos (textos ou blocos) o contornem.

- LEFT – usando este valor ele irá indicar que o determinado bloco flutue à esquerda. Exemplo:

```
.produto .produto-imagem {  
  float: left;  
}
```

- LEFT – usando este valor ele irá indicar que o determinado bloco flutue à esquerda. Exemplo:

```
.produto .produto-descricao {  
  float: right;  
}
```

- NONE – o elemento não flutua;
- INHERIT – indica que irá herdar o float do float do elemento pai caso o mesmo esteja declarado. Exemplo:

```
.produto {  
  float: left;  
}  
.produto .produto-imagem {  
  float: inherit;  
}
```

- CLEAR – não é nenhum valor atribuído ao float e sim, uma propriedade que indica uma “quebra de linha”, digamos assim. Ao usar o float para indicar se o elemento irá flutuar para a esquerda ou direita, caso precise indicar que um outro bloco ficará abaixo dos que estão acima, usa-se *clear:both*; Exemplo:

```
.produto .produto-imagem {  
  float: left;  
}  
  
.produto .produto-descricao {  
  float: right;  
}  
  
.produto .preco {  
  clear: both;  
}
```

```
<body>  
  <div class="produto">  
    <div class="produto-imagem">  
    </div>  
    <div class="produto-descricao ">  
    </div>  
    <div class="produto-preco">  
    </div>  
  </div>  
</body>
```

- BOTH – é o mais usado e indica que o conteúdo deve sempre permanecer abaixo dos elementos flutuantes, seja à esquerda ou direita.

Propriedade Position: Ao contrário do que se possa imaginar, a propriedade position não é indicada para se trabalhar com estrutura de layouts. Pelo contrário, ele serve para posicionar pequenos elementos, detalhes no layout. Para trabalhar os posicionamentos dos blocos usa-se o float e atualmente, o flexbox (que, em breve, publicarei um conteúdo extra falando somente da propriedade display:flex e como usá-lo). O position tem 4 valores:

- STATIC – ele é o valor padrão e vai seguir o estrutura da página HTML.

```
.conteudo {  
  position: static;  
}
```

- STATIC – ele é o valor padrão e vai seguir o estrutura da página HTML.

```
.conteudo {  
  position: relative;  
}
```

- ABSOLUTE – através desse valor é possível posicionar qualquer elemento de acordo com o elemento pai usando qualquer outra position sem ser o static. O mais comum é usar o pai como relative e filho como absolute. Exemplo:

```
.conteudo {  
  position: relative;  
}  
.conteudo .destaque {  
  position: absolute;  
  top: 10px;  
  bottom: 20px;  
  right: 10px;  
  left: 10px;  
}
```

- FIXED – como o próprio nome já diz, a posição do elemento é fixo. Mesmo possuindo um comportamento similar ao absolute, ele especifica a posição do elemento em relação à janela do seu browser. Mesmo com o scroll, o elemento permanece fixo.

```
.conteudo .menu {  
  position: fixed;  
  top: 0;  
}
```

Label: termo em inglês para "etiqueta", "rótulo".

Responsividade: sites responsivos são aqueles que adaptam o tamanho das suas páginas (alteração do layout) ao tamanho das telas que estão sendo exibidos, como as telas de celulares e tablets. Suas vantagens derivam da adaptação a qualquer ferramenta que os usuários estejam usando para facilitar a sua visualização.



JavaScript: linguagem de script para controlar e criar conteúdo dinâmico de um site, sem que seja necessário recarregar a página toda vez que algo mudar.