

DESENVOLVIMENTO DE SOFTWARE PARA WEB

CAPÍTULO 1 - POR ONDE COMEÇAMOS, PARA CRIAR PÁGINAS HTML?

Fernando Cortez Sica

Introdução

Para começar a compreender como funciona o desenvolvimento de softwares para a web, precisamos saber, antes de tudo, como se desenvolve uma página web. Mas, páginas web são construídas com qual linguagem? Há uma linguagem básica, que é a primeira que vamos estudar, para que possamos implementar páginas web, que é a Linguagem de Marcação de Hipertexto, conhecida por HTML (*HyperText Markup Language*).

Após as conceituações iniciais sobre HTML, serão apresentados os elementos básicos para a criação de tabelas e formulários, que são elementos muito importantes para as páginas, pois permitem uma organização de conteúdo e uma primeira forma de interação com o usuário da página. Mas, como realizar o alinhamento das informações dentro das tabelas? Abordaremos esse tema aliado ao tema de mesclagem de tabelas. Assim, as páginas ficarão mais funcionais e estruturadas.

E os formulários? São muito difíceis de serem manipulados? Falaremos sobre isso quando dialogarmos sobre as estruturas e os campos de um formulário. Dessa forma, você terá amplas condições de criar suas primeiras páginas web.

Mas, existe alguma outra forma de formatar uma página HTML? Sim, ao final do capítulo será abordado o tema CSS (*Cascading Style Sheets*, traduzido como Folhas de Estilo em Cascata). Para adiantar nossa conversa, o CSS serve para configurar a aparência dos elementos descritos com HTML. Uma outra forma para a criação de páginas HTML consiste em utilizar um *framework*. Mas, o que vem a ser um *framework*? Falaremos sobre isso quando conversarmos sobre o Bootstrap.

A visão que você terá ao fim deste capítulo permitirá que você desenvolva suas primeiras páginas. Vamos começar? Bons estudos!

1.1 Desenvolvimento de páginas web com HTML

Antes de começar a falar sobre HTML, seria interessante explicar um pouco sobre a origem do nome HTML (*HyperText Markup Language*, traduzido como Linguagem de Marcação de Hipertexto).

Mas, o que vem a ser hipertexto? Um hipertexto caracteriza-se por um texto no qual pode-se encontrar *links* para permitir uma leitura não linear de seu conteúdo (JOÃO, 2014). Pelos *links*, o leitor terá a possibilidade de navegar em outras páginas, inclusive de outros domínios da internet.



Fonte: Biz Idea Production, Shutterstock, 2018.

Os navegadores da internet, como o *Google Chrome* e o *Internet Explorer*, manipulam, dentre outros conteúdos, código HTML. Para permitir o intercâmbio de conteúdo, os navegadores são implementados seguindo o protocolo de rede conhecido como HTTP e HTTPS (*HyperText Transfer Protocol* e *HyperText Control Protocol Secure*, respectivamente, Protocolo de Transferência de Hipertexto e Protocolo de Transferência de Hipertexto Seguro).



http vs https

Fonte: Teguh Jati Prasetyo, Shutterstock, 2018.

Após essa breve introdução, chegou a hora de iniciar a exploração do HTML, começando com a sua estruturação e principais *tags*.

1.1.1 Estrutura *HTML*

Para entender a estrutura do código HTML, vamos conversar um pouco da estrutura das páginas que você criar. As páginas deverão ter um conteúdo claro e objetivo, além de bem organizado. As páginas a serem criadas deverão funcionar na maioria dos navegadores, ou seja, não se deve pensar em implementá-las em um tipo de navegador específico. Sendo assim, os testes de uma página devem ser feitos em vários navegadores, para que se possa verificar possíveis distorções na aparência e, em alguns casos, na própria exibição do conteúdo.

VOCÊ SABIA?



Você sabia que o *layout* das páginas, assim como, por exemplo, a utilização das cores é um estudo das áreas de semiótica e IHC (Interface Humano-Computador)? Estudos devem ser realizados para que a página possa prover maior usabilidade. Para saber mais, acesse o artigo “O que é Usabilidade?” (PAGANI, 2011), disponível em <https://tableless.com.br/o-que-e-usabilidade/>.

A princípio, a codificação HTML baseia-se na utilização de *tags*. *Tags* são marcadores que delimitam uma região do texto para que seja aplicada alguma regra de configuração (MILETTO; BERTAGNOLLI, 2014). Desta forma, a estruturação geral de uma *tag* HTML pode ser observada na figura a seguir.

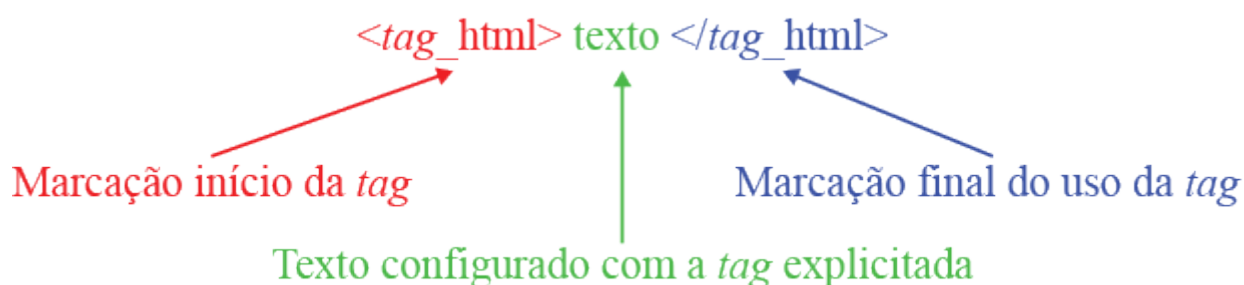


Figura 1 - Estruturação genérica para a utilização das tags. Nota-se que o texto deverá ficar entre a marcação de início e a marcação de final da tag específica.

Fonte: Elaborada pelo autor, 2018.

A figura acima ilustra que o conteúdo (denotada, na figura, como “texto”) deve estar delimitado pela marcação de início e de final da *tag*. Desta forma, o navegador tem condições de delinear, em seu processamento, qual a região que será impactada pela *tag* em questão. Convém, aqui, relatar que a marcação de finalização da *tag* sempre apresenta o mesmo nome em relação à marcação de início, com a diferença que apresenta o uso da barra (“/”). O HTML apresenta, em sua definição, um amplo conjunto de *tags* associadas com diversas funcionalidades. Para o nosso início de conversa, veremos, a seguir, alguns exemplos de *tags* importantes para a definição da estrutura geral da página e para a elaboração de sua primeira página baseada em HTML.

1.1.2 Principais tags

Para utilizar as *tags* para codificar uma página em HTML, precisamos definir prioridades. As primeiras *tags* que poderemos utilizar são aquelas que objetivam a definição da estrutura da página e nomes dos títulos, cabeçalhos e parágrafos.

As *tags* que definem a estrutura da página não impactam na formatação e, consequentemente, a forma de exibição da página, elas são apenas informativas (LEMAY; COLBURN; TYLER, 2002). Para sermos mais específicos, a página pode ser estruturada por intermédio da utilização das *tags* <HTML>, <head> e <body>. Clique nos *cards* abaixo e entenda cada uma dessas *tags*.

Tag <HTML>

Esta *tag* apenas indica que o documento foi criado usando-se a linguagem HTML. Sendo assim, qualquer página HTML sempre apresenta a *tag* <HTML> no início de sua codificação.

Tag <head>

A *tag* <head> tem a função de delimitar o cabeçalho da página, ou seja, nessa seção poderão conter, por exemplo, o título e o autor da página em questão. Neste contexto, você poderá usar as *tags* <title> e “*meta tags*”, respectivamente.

Tag <body>

A utilização desta *tag* serve para delimitar o conteúdo da página propriamente dito. Entre a marcação de início (<body>) e a marcação de finalização (</body>), encontram-se, por exemplo, o texto, os *links* e as tabelas da página.

A figura abaixo ilustra a utilização das *tags* <HTML>, <head>, <title> e <body> além de exemplos de uso das *meta tags*.

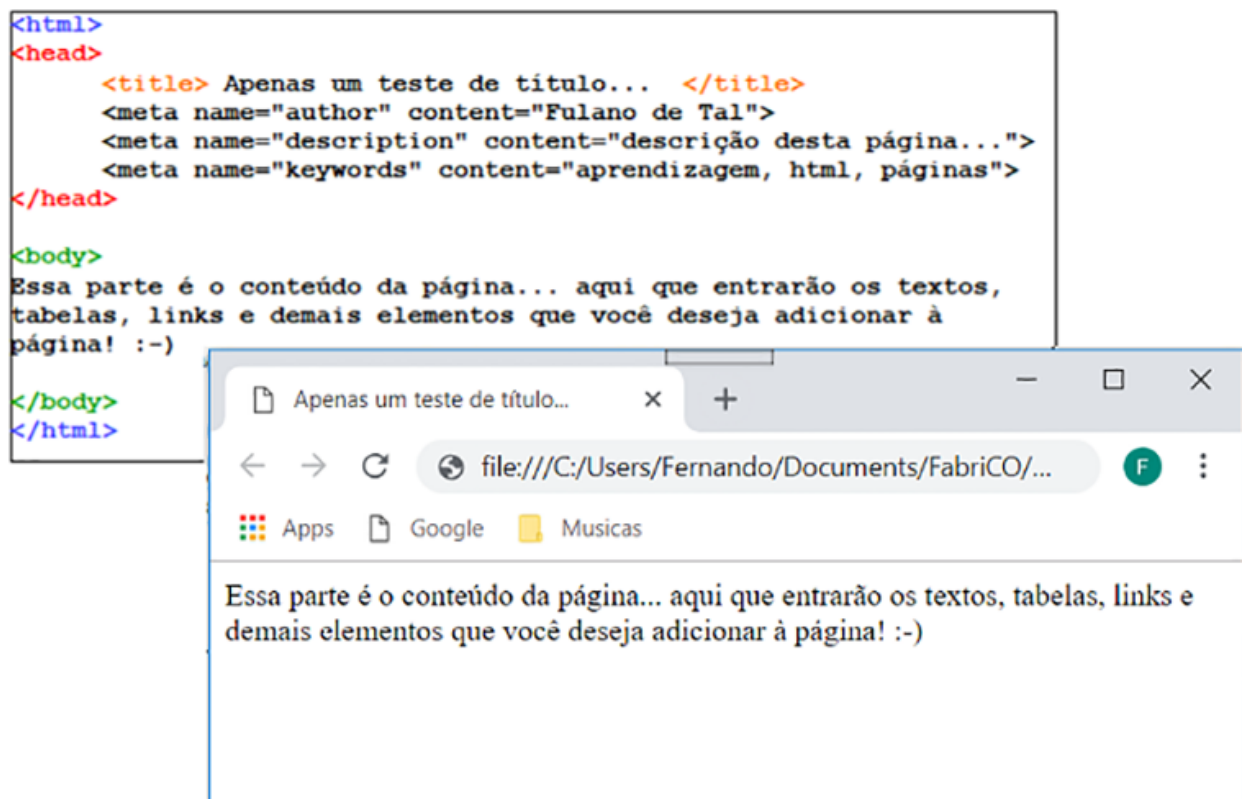


Figura 2 - Exemplo de utilização das tags "HTML", "head", "title" e "body" além de exemplos de uso das “meta tags”.

Fonte: Elaborada pelo autor, 2018.

Olhando a figura anterior, percebemos que as marcações de início e de finalização estão alinhadas, ou seja, a primeira marcação que aparece é fechada por último – as marcações de encerramento aparecem na ordem inversa, em relação às marcações de início. Você deve estar se perguntando: “e as *meta tags*?”

As *meta tags* tem várias funcionalidades, dentre as quais podemos citar, inicialmente, a inserção de informações para as máquinas de busca. Sendo assim, por meio da *meta tag*, por exemplo, *keywords* (palavras-chave), a máquina de busca poderá associar a sua página à *query* de busca solicitada por algum usuário. A manipulação da *meta tag* foi feita, neste caso, definindo-se o nome do campo a ser instanciado (denotado pela palavra “*name*” – nome) e o valor a ser associado ao referido campo (denotado por “*content*” – conteúdo).

VOCÊ QUER LER?



Como as *meta tags* são utilizadas para passar informações aos mecanismos de busca (como o Google e o Bing), é por elas que sua página pode ser encontrada como resposta a uma consulta. Para saber mais sobre *meta tags*, recomendamos o artigo “Código Tag - O que é *meta tag*?” (ARRIGONI, 2012), disponível em: <http://www.linhadecodigo.com.br/artigo/3595/codigo-tag-o-que-e-meta-tag.aspx>.

Você pode, ainda, formatar o seu corpo, utilizando *tags* que modificam o tamanho da fonte a ser exibida. Tais *tags*, compreendendo seis níveis, são denominadas como *tags* de cabeçalhos – <h1> até <h6> (LEMAY; COLBURN; TYLER, 2002).

Para organizar melhor seu texto, você pode contar com as outras *tags*. Clique nos ícones abaixo e confira.

Tag de parágrafo <p>: para visualizar melhor seu texto, você pode dividi-lo em parágrafos delimitados pela marcação de início <p> e pela marcação de finalização </p>.

Tags de lista e : você pode criar listas não ordenadas, ou ordenadas usando as *tags* (*unordered list* – lista não ordenada), ou (*ordered list* – lista ordenada), respectivamente.

Nas *tags* de lista, cada item da lista deve estar delimitado por uma das seguintes *tags*:

- <dt> ... </dt>: o texto do item aparece sem nenhum marcador (*bullet*);
- <dd> ... </dd>: com o uso desta *tag*, o item aparece com um nível a mais de indentação. No contexto de programação, indentar significa alinhar os blocos de linhas de código para que as suas interdependências fiquem mais evidenciadas;
- ... : caso você queira que cada item seja precedido por um marcador (*bullet*), você pode fazer uso desta *tag*.

Na figura a seguir, temos um exemplo de utilização das *tags* de formatação, vistas até o momento. Perceba que estamos fazendo um incremento da página construída anteriormente pela adição das *tags* de formatação de texto. Por conta do espaço, os elementos inseridos na figura anterior não foram repetidos no código HTML.

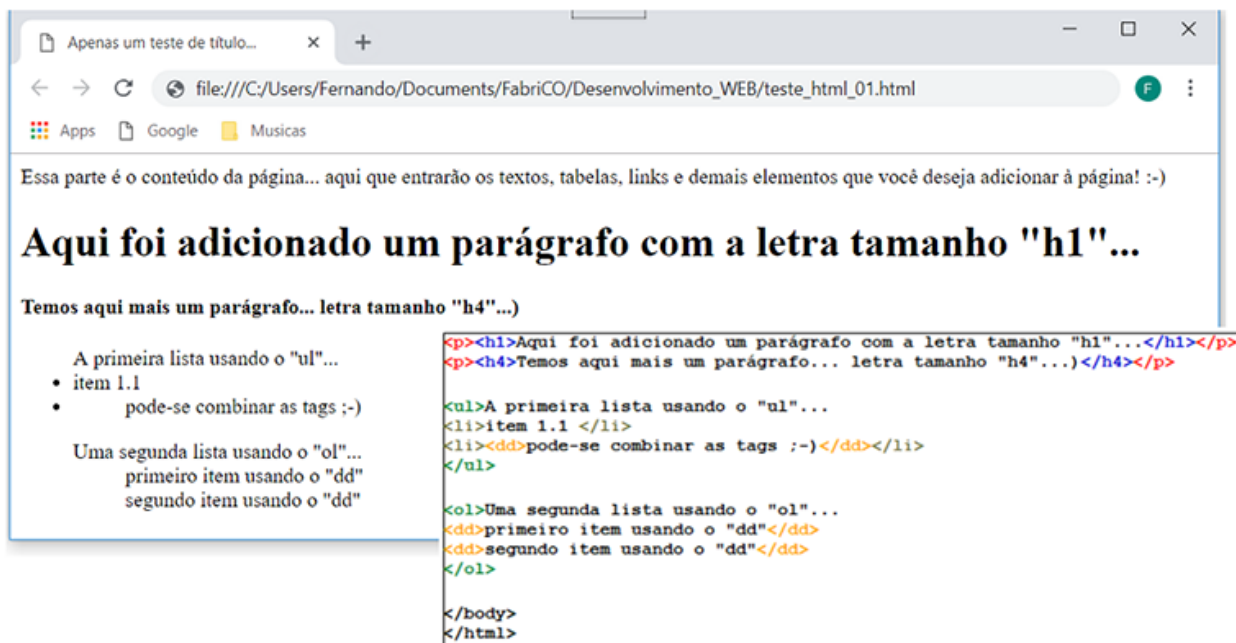


Figura 3 - Exemplo de utilização das tags básicas de formatação de texto e de criação de listas não ordenada e ordenada.

Fonte: Elaborada pelo autor, 2018.

A figura acima mostra que as *tags* poderão ser utilizadas em conjunto, ou seja, um certo item, ou texto pode ser formatado por várias *tags* usadas em conjunto. Neste caso, deve-se prestar atenção na ordem da colocação dos marcadores de finalização – devem aparecer na ordem inversa, em relação aos marcadores de início.

Você poderá, ainda, usar outras formatações para as listas ordenadas e não ordenadas. Para tanto, usa-se os formataadores “*type*” e “*start*”.

Como exemplo de utilização para lista ordenada, temos:

```
<ol type="I" start="5">... </ol>
```

Por vez, para uma lista não ordenada, podemos usar:

```
<ul type="circle">... </ul>
```

Estes dois exemplos produzirão, respectivamente, uma marcação em itálico iniciando-se no valor 5 e a utilização de marcador no formato de uma circunferência.

VOCÊ O CONHECE?



O cientista da computação inglês Tim Berners-Lee foi o criador da *World Wide Web* (WWW), proposta em 1989, como um sistema para integrar redes menores. Na época, Berners-Lee era engenheiro de *software* da CERN (Organização Europeia para Pesquisa Nuclear), cuja sede, em Genebra, recebia visitas de cientistas do mundo inteiro. Ele percebeu a dificuldade que havia para acessar os dados de pesquisas e pensou em uma linguagem de marcadores (HTML), a URL e o protocolo HTTP, para possibilitar o compartilhamento remoto de informações (FÁBIO, 2018). Leia mais em: <https://www.nexojornal.com.br/expresso/2018/03/13/Tim-Berners-Lee-o-criador-da-internet-global.-E-suas-cr%C3%ADticas-%C3%A0-rede-hoje>.

Uma outra observação que merece ser mencionada, consiste no fato de que é possível criar uma lista dentro de outra. Dessa forma, consegue-se alinhar os elementos a serem exibidos em sua página. Sobre as listas e outras *tags* de formatação, os pesquisadores Lemay, Colburn e Tyler (2002) e Miletto e Bertagnolli (2014) são os mais indicados para os estudos.

Até este momento, você pode criar páginas simples contendo, em seu conteúdo, apenas textos básicos. Nas próximas seções, vamos aprender a incorporar mais elementos à página, iniciando com a inserção de tabelas.

1.2 Criando tabelas em HTML

Além dos textos e listas já mencionados, uma importante estrutura que você pode inserir em suas páginas consistem nas tabelas. Com as tabelas, você poderá organizar melhor o conteúdo de sua página.

Pelas tabelas, você poderá fazer com que o seu texto apareça, por exemplo, em uma coluna ao lado de uma figura. Para a criação de tabela, você deverá, antes, pensar na diagramação de sua página, ou seja, qual será o formato de sua tabela, bordas, coloração, conteúdo das células etc. Essa parte é de suma importância para que você possa estruturar o seu código, baseado no *layout* esquematizado.

Clique nas abas abaixo e veja as partes que uma tabela apresenta, segundo Lemay, Colburn e Tyler (2002).

-

Título da tabela

Corresponde ao nome da tabela – item opcional quando, por exemplo, a sua tabela servir para organizar e distribuir o conteúdo na página.

-

Cabeçalhos das linhas e colunas

Nomes dados às linhas e/ou às colunas para explicar, ao leitor da página, o que representa o conteúdo daquela linha e/ou coluna.

-

Células da tabela

São os próprios espaços delimitados (com ou sem borda) da tabela.

-

Dados

Representam os próprios conteúdos das células.

Para a criação da tabela, usa-se a *tag* de marcação de início `<table>` e a sua correspondente marcação de finalização `</table>`. No momento da criação da tabela, podemos, também, configurar, por exemplo, borda, dimensão da tabela e dimensão de preenchimento da célula (W3SCHOOLS, [s/d]) (LEMAY; COLBURN; TYLER, 2002). Para tanto, usamos os parâmetros associados no momento da criação da tabela da seguinte forma, por exemplo:

```
<table border="1" style="width:50%" cellpadding="10" cellspacing="6">.
```

Acompanhe na interação a seguir, a explicação deste exemplo.

Foi definida uma borda de largura de 1 *pixel* para a borda da tabela (pelo parâmetro *border* – borda). Você poderá testar outros números para a largura da borda e observar as suas variações.

Por sua vez, o parâmetro identificado por *style-width* (estilo-largura) referencia ao espaço que ocupará na página. Neste exemplo, foi definido que a tabela ocupará apenas 50% da largura total da página. Caso seja definida uma largura insuficiente para exibir o conteúdo da tabela, o navegador, automaticamente, adequará a largura para que seja possível a exibição de todo o conteúdo das células.

Já o parâmetro *cellpadding* (preenchimento da célula) refere-se à distância (em *pixels*) entre a borda e o conteúdo da célula. Por exemplo, caso o valor associado seja 0 (zero), não haverá espaço entre a borda e o conteúdo da célula.

Por fim, o *cellspacing* (espaçamento da célula) denota, em *pixels*, o espaçamento entre as células. No caso, foi criado um espaçamento de 6 *pixels* entre as bordas e cada célula.

Voltando às **partes da tabela**, referenciadas a pouco, clique nas abas abaixo e veja que elas poderão ser explicitadas pelas seguintes *tags*:

Ⓔ

Título: como já mencionamos, o título é opcional. Mas, caso você necessite inseri-lo, pode usar a seguinte forma: `<caption>Aqui será inserido o título da tabela</caption>`.

Ⓕ

Conteúdo das células: cada linha da tabela será indicada e delimitada pelas *tags* de marcação `<tr>... </tr>` (*tr* = *table row* – linha da tabela). Dentro de cada linha, serão inseridos os conteúdos das células, sendo que podemos diferenciar aqueles que são cabeçalhos frente aos dados efetivos. A marcação dos cabeçalhos é feita pelo par `<th>... </th>` (*th* = *table header* – cabeçalho da tabela) e, os dados devem estar envolvidos por `<td>... </td>` (*td* = *table data* – dado da tabela). Caso seja necessária uma célula vazia, basta inserir: `<tr></td>`, ou usar uma quebra de linha: `<tr>
</tr>`.

Assim como podemos definir o **tamanho da tabela** como um todo, é possível definir o tamanho das células passando o parâmetro *width* à *tag* `<td>` ou `<th>` da seguinte forma: `<td width="25%">Conteúdo da célula...</td>`

Além de configurar distâncias, bordas e demais elementos que nós acabamos de ver, podemos, ainda, mudar a **configuração das cores de preenchimento da tabela e das células, assim como as cores das bordas**. Para tanto, basta fazer uso do parâmetro *bgcolor* (*background color* – cor de fundo) dentro das *tags* de marcação de início `<table>`, `<tr>`, `<th>` ou `<td>`. Desta forma, no exemplo:

`<td width="25%" bgcolor="red"> conteúdo da célula </td>`

temos a exibição do texto sobre um plano de fundo vermelho (*red*, em inglês). Como você notou, a passagem do parâmetro relativo à cor pode ser feita por meio da referência ao próprio nome da cor, ou por um número hexadecimal que represente a cor em sua composição RGB, sendo R=*Red* (vermelho); G=*Green* (verde); B=*Blue* (azul). Por exemplo, para o vermelho, basta ativar o máximo valor no campo “R”: #FF0000; o verde corresponde a #00FF00 e, por último, o azul vale #0000FF. O sinal “#” indica que o valor escrito está no formato hexadecimal.

CASO

Para o desenvolvimento de uma página, são necessárias várias etapas, desde o esboço inicial da página (traçada após a primeira conversa com o cliente) até a colocação da página no ar. Um aspecto importante para o sucesso de uma página, consiste em criar um *layout* que seja facilmente navegável e que, visualmente, seja agradável não somente perante aos olhos mas, também, por exemplo, perante a movimentação do *mouse* sobre a tela, a fim de alcançar os pontos de clique e de preenchimento de campos de formulário. Esses levantamentos são realizados pelo *briefing* (análise de perfil, dentre outros, do mercado e dos possíveis usuários) e por meio de objetivos do próprio cliente. Após os primeiros protótipos, várias alterações (realização de ações de inclusão, retirada, reposicionamento e substituição de componentes) foram necessárias, de forma a facilitar a aprendizagem de utilização do *site* pelo usuário e otimizar o número de *clicks* para chegar aos objetivos. O artigo “Processo de criação em *websites*: um estudo de caso do *site* Ilesam” (ASSUMPÇÃO; VILLEGAS, 2013) retrata um estudo de caso relacionado ao processo de criação de *websites*. O referido estudo de caso poderá ser acessado pelo link <<https://periodicos.utfpr.edu.br/de/article/download/2129/2028>>.

Da mesma forma que você pode alterar a cor do plano de fundo, você poderá testar a mudança de cor da fonte da seguinte forma:

```
<td width="25%" bgcolor="red" style="color:#0000FF">conteúdo</td>.
```

Nesse exemplo, foi escrito o texto na cor azul (configurado pelo campo `style="color:#0000FF"`) sobre o fundo vermelho (`bgcolor="red"`).

Ainda falando sobre coloração, alterações nas cores das bordas podem não ser processadas por alguns navegadores. Neste caso, sugerimos o uso de CSS, que vamos estudar mais adiante neste capítulo.

Até o momento, conversamos sobre criação da tabela e formatação das células. Mas, tem como alinhar o conteúdo das células? A seguir, abordaremos esse assunto.

1.2.1 Alinhamento

Alinhar o conteúdo é importante pela questão estética, aliada à organização da página em relação ao *layout*. É possível realizar o alinhamento da tabela de cada célula. Mas, você deve estar se perguntando: alinhar uma tabela significa configurar os alinhamentos de todas as células de uma vez só?

Existe uma diferença – alinhar uma tabela significa definir a sua posição dentro da página, ou seja, a tabela ficará centralizada na página, junto à margem direita, ou junto à margem esquerda? Clique nas setas da interação abaixo e entenda melhor.

Em ambos os casos, tabela, ou cada célula, você poderá realizar o alinhamento com o parâmetro *align* (alinhar) no momento da definição da tabela, do título, da linha, do cabeçalho ou de cada célula (*tags* `<table>`, `<caption>`, `<tr>`, `<th>` ou `<td>`, respectivamente).

Alinhar uma linha significa que todas as suas células serão impactadas pela configuração. Caso seja necessária uma configuração particular de uma célula em específico, deve-se colocar o parâmetro *align* no momento da criação do item da célula (seja cabeçalho, ou dado).

O parâmetro *align* aceita, como valores possíveis: *left* (esquerda), *right* (direita), ou *center* (centralizado). Sendo assim, uma possível utilização do *align* seria: `<tr align="center">`.

Mas, podemos alinhar somente na horizontal? Seria possível uma atuação na vertical? Sim, podemos alinhar, também, na vertical usando o parâmetro *valign* (*vertical align* = alinhamento vertical). Para tanto, você poderá passar os valores *top*, *bottom*, *center* para o *valign*.

Por falar em alinhamento, todas as células deverão ter o mesmo tamanho dentro de sua coluna ou de sua linha? Para responder a esta questão, abordaremos, a seguir, o tema de tabela mesclada.

1.2.2 Mesclagem

Você sabe o que seja uma tabela mesclada? Tabela mesclada (ou realizar mesclagem na tabela) consiste em unir duas ou mais células pertencentes a uma mesma linha, ou pertencentes a uma mesma coluna.

Uma das funcionalidades de se mesclar células consiste em, por exemplo, permitir a criação de cabeçalhos que representam uma classe que engloba vários itens. Por exemplo, suponha que você deseja criar uma tabela para disponibilizar o balanço financeiro. Dessa forma, você pode criar, por exemplo, uma célula que possua um item para os “débitos”, englobando, dessa forma, células subordinadas – explicitando o tipo do débito (tais como: aluguel, transporte, alimentação etc.).

Pelo fato de envolver somente células, a mesclagem apenas se aplica às *tags* `<th>` e `<td>`. As junções se estendem sempre da esquerda para a direita (no caso de junções dentro da mesma linha), ou de cima para baixo (quando forem unidas células de uma mesma coluna) (LEMAY; COLBURN; TYLER, 2002).

Mas, como unir as células para criar as chamadas células mescladas? Clique nos botões abaixo e entenda.

- - 1
Para criar células que envolva várias colunas adicione, à tag `<th>`, ou `<td>`, o parâmetro `colspan` – informando o número de colunas abrangidas.
- - 2
Por sua vez, a mesclagem de células que cobrem várias linhas é realizada pelo parâmetro `rowspan`.
- - 3
Sendo assim, por exemplo, para criar uma célula com o tamanho relativo a duas colunas, usa-se:
`<td colspan="2">Conteúdo da célula</td>`.

Para sintetizar algumas *tags* apresentadas na interação anterior, a figura a seguir ilustra a utilização das *tags* para a criação de uma tabela, formatação das células, alinhamento e mesclagem.

```

<table cellpadding="10" align="center" bordercolorlight="Red"
bordercolordark="yellow" border="8" style="width:50%">
<tr>
<th style="border: 0px solid #000;"><br /></th>
<th bgcolor="#808080" align="center" colspan="3">Meses</th>
</tr>
<tr>
<th style="border: 0px solid #000;"><br /></th>
<th bgcolor="#808080" width="25%" style="color:#0000FF">Janeiro</th>
<th bgcolor="#808080" width="25%" style="color:#0000FF">Fevereiro</th>
<th bgcolor="#808080" width="25%" style="color:#0000FF">Março</th>
</tr>
<tr>
<th>Vendas Produto (A)</th>
<td>10</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<th>Vendas Produto (B)</th>
<td>8</td>
<td>3</td>
<td>9</td>
</tr>
</table>

```

	Meses		
	Janeiro	Fevereiro	Março
Vendas Produto (A)	10	15	7
Vendas Produto (B)	8	3	9

Figura 4 - Exemplo de utilização das tags para a criação de uma tabela, formatação das células, alinhamento e mesclagem.

Fonte: Elaborada pelo autor, 2018.

Na figura acima, percebemos a aplicação das *tags* referenciadas nesta seção. Uma observação que merece destaque consiste no fato de que o programador também pode configurar bordas de uma célula. No caso do exemplo acima, temos, por exemplo, no momento da criação do cabeçalho em branco, a redefinição da borda para, apenas, esta célula em específico. No caso, definiu-se como uma célula cuja borda possui a largura de 0 *pixel* (0 px) e cor branca (solid #000).

VOCÊ SABIA?




Você sabia que existem “tabelas acessíveis”? A acessibilidade é uma preocupação constante na atualidade. As tabelas HTML não fogem a essa preocupação (QUEIROZ; PORTA, 2009). Para saber sobre “tabelas acessíveis”, leia mais em: <<http://www.acessibilidadelegal.com/13-tabelas-acessiveis.php>>.

Até o momento, falamos apenas sobre formas de exibir informações estáticas. Mas, você pode notar que, na internet, as páginas permitem, também, interação. Sendo assim, o leitor da página pode inserir informações para que sejam transmitidas ao domínio da página em questão. Essa inserção é realizada por meio de formulários, que serão abordados a seguir.

1.3 Criando formulários em HTML

Como o usuário interage com um sistema por trás de uma página? Um primeiro modo para permitir essa interação é o uso de formulários. Sendo assim, formulários HTML são elementos HTML capazes de fornecer ao usuário condições para que ele forneça informações sob a forma, por exemplo, caixa textual (*text box*), caixa de seleção (*combo box*), caixa de verificação (*checkbox*) e botões de rádio (*radio buttons*).



First Name

Last Name

Username

Email

City

Address

Phone Number

Create Account

Fonte: Anastasia_B, Shutterstock, 2018.

Basicamente, as informações fornecidas por um formulário podem seguir por caminhos diferentes, como por exemplo: com a ativação de um botão, inicia-se o processo de envio de *e-mail* para uma determinada pessoa (ou o próprio usuário); pelos formulários, o usuário é direcionado para alguma página específica; ativa-se um *script* para que a informação seja processada no servidor (LEMAY; COLBURN; TYLER, 2002). Para esse atendimento,

diversas são as tecnologias que poderemos utilizar, dentro das quais, podemos destacar: CGI (*Common Gateway Interface* – interface de passagem comum), FastCGI e WSGI.

VOCÊ SABIA?



Você sabia que o CGI é uma tecnologia considerada como obsoleta? Pelo fato de que a cada requisição ao servidor, cria-se um processo novo, essa tecnologia não atende às expectativas de sistemas maiores, com um grande número de requisições. Quer saber mais? Acesse o artigo “Entendendo o CGI, FastCGI e WSGI”, que descreve o FastCGI e o WSGI como alternativas ao CGI (LAUBE, 2012). Disponível em <<https://klauslaube.com.br/2012/11/02/entendendo-o-cgi-fastcgi-e-wsgi.html>>.

Todo formulário é marcado pelas *tags* `</form>... </form>`. Mas, qual o código a ser inserido entre as *tags* de marcação? Veremos isso agora, conversando sobre estruturas e campos dos formulários.

1.3.1 Estruturas e campos

Para começar, vamos supor a necessidade de criar um formulário para que usuário forneça seus **dados pessoais** (como nome, endereço, Estado etc.). Como primeiro contato em relação à manipulação de formulários, não faremos, por exemplo, validação das informações digitadas. Para exemplificar o código, foi referenciado um *script* hipotético, usando a linguagem PHP (*Preprocessor Hypertext* – pré-processador hipertexto).

Para facilitar a nossa conversa, apresentamos, na figura abaixo, o código do formulário que iremos trabalhar no começo para que, depois, possamos explicar as principais linhas.


```

<!-- Numeração das linhas como comentários -->
<!-- 01 --> <html>
<!-- 02 -->   <head>
<!-- 03 -->   <title> Testando formulários... </title>
<!-- 04 -->   <meta name="description" content="Criação de formulários HTML">
<!-- 05 -->   <meta http-equiv="Content-Type" >
<!-- 06 -->   </head>
<!-- 07 -->   <body>
<!-- 08 -->   <h1> Formulário para dados pessoais</h1>
<!-- 09 --> <form action="DadosPessoais.php" method="post">
<!-- 10 -->   <fieldset>
<!-- 11 -->   <legend>Informações do Usuário</legend>
<!-- 12 -->   <table cellpadding="8">
<!-- 13 -->   <tr>
<!-- 14 -->       <td>
<!-- 15 -->       <label for="nome">Nome: </label>
<!-- 16 -->       </td>
<!-- 17 -->       <td align="left">
<!-- 18 -->       <input type="text" name="nome" value="Digite o seu nome">
<!-- 19 -->       </td>
<!-- 20 --> </tr>
<!-- 21 -->   <td>
<!-- 22 -->   <label>Data de nascimento: </label>
<!-- 23 -->   </td>
<!-- 24 -->   <td align="left">
<!-- 25 -->       <input type="text" name="dia" size="2" maxlength="2" value="dd">
<!-- 26 -->       <input type="text" name="mes" size="2" maxlength="2" value="mm">
<!-- 27 -->       <input type="text" name="ano" size="4" maxlength="4" value="aaaa">
<!-- 28 -->       </td>
<!-- 29 -->   </tr>
<!-- 30 -->   <tr>
<!-- 31 -->       <td>
<!-- 32 -->       <label for="Reg">Região:</label>
<!-- 33 -->       </td>
<!-- 34 -->       <td align="left">
<!-- 35 -->       <select name="Reg">
<!-- 36 -->       <option value="N">Norte</option>
<!-- 37 -->       <option value="NE">Nordeste</option>
<!-- 38 -->       <option value="CO">Centro-oeste</option>
<!-- 39 -->       <option value="SE">Sudeste</option>
<!-- 40 -->       <option value="S">Sul</option>
<!-- 41 -->       </select>
<!-- 42 -->       </td>
<!-- 43 -->       <td>
<!-- 44 -->       <label for="senha">Senha:</label>
<!-- 45 -->       </td>
<!-- 46 -->       <td align="left">
<!-- 47 -->       <input type="password" name="senha">
<!-- 48 -->       </td>
<!-- 49 -->   </tr>

```

```

<!-- 50 --> </table>
<!-- 51 --> </fieldset>
<!-- 52 --> <input type="submit" value="Enviar">
<!-- 53 --> <input type="reset" value="Limpar Formulário">
<!-- 54 --> </form>
<!-- 55 --> </body>
<!-- 56 --> </html>

```

Figura 5 - Exemplo de código HTML para a criação de formulário. No caso, estão sendo utilizados campos textuais, de seleção, para senha e botões.

Fonte: Elaborada pelo autor, 2018.

Como mencionado, vamos analisar o código da figura acima, para entender um pouco a manipulação de formulários. Clique nos botões abaixo.

A definição do formulário inicia na linha 9, junto à *tag* de marcação `<form>`.

```
<!-- 09 --> <form action="DadosPessoais.php" method="post">
```

form

Nesta linha, notamos dois campos relativos ao formulário: atributo *action* (ação) e atributo *method* (método). Ao definir *action* o *script* receberá os dados do formulário para que haja o processamento. Para tanto, pode ser referenciado um arquivo de código local, ou associar uma URL (*Uniform Resource Locator* – Localizador Uniforme de Recursos) que contém o *script*. O referido *script* receberá os dados fornecidos pelo formulário, assim que o botão *submit* (linha 52) for pressionado.

Ainda em relação à linha 9, o atributo *method* define a forma de passar a informação a ser processada.

```
<!-- 09 --> <form action="DadosPessoais.php" method="post">
```

method

A diferença está na forma de envio e visibilidade – enquanto que, na forma *get*, as informações são passadas junto ao endereço de destino – neste caso, as informações ficam visíveis pelo fato de que elas compõem o caminho completo da URL. Por sua vez, na forma *post*, as informações são encapsuladas e enviadas pelo *HTTP post* – neste caso, as informações são mais seguras, não visíveis no navegador (VICTORIO, 2016).

A *tag fieldset* (linha 10) é responsável por agrupar os campos do formulário, criando uma borda em torno deles.

```
<!-- 10 --> <fieldset>
```

fieldset e legend

Essa *tag* pode ser complementada pela *tag legend* (legenda), presente na linha 11, para que seja exibida uma legenda associado ao bloco de campos agrupados.

	<pre><!-- 11 --> <legend>Informações do Usuário</legend></pre> <p>As linhas 12, 13 e 14 contêm elementos que já lhe são conhecidos: a definição de tabela (<code><table></code>), marcação de início de uma linha da tabela (<code><tr></code>) e definição do conteúdo de uma célula (<code><td></code>).</p>
table	<pre><!-- 12 --> <table cellpadding="8"> <!-- 13 --> <tr> <!-- 14 --> <td></pre> <p>Passando para a entrada efetiva da informação, temos, na linha 18, a <i>tag</i> relacionada à entrada de dados do formulário: <code><input></code>.</p>
input	<pre><!-- 18 --> <input type="text" name="nome" value="Digite o seu nome"></pre> <p>Nessa linha, foi definida uma entrada textual (<code>type="text"</code>) – diferentemente das entradas realizadas nas linhas 35 (onde a entrada é feita por uma seleção – a ser visto mais adiante), linha 47 (a informação digitada não aparecerá diretamente na tela, ou seja, serão exibidos marcadores para a entrada da senha) e, finalmente, linhas 52 e 53, cujas entradas representam botões.</p>
name e value	<p>Continuando na linha 18, temos o campo <i>name</i> (nome) cuja função é criar um identificador a ser passado para o <i>script</i>.</p> <pre><!-- 18 --> <input type="text" name="nome" value="Digite o seu nome"></pre> <p>Na programação do <i>script</i>, deve aparecer uma variável com o mesmo nome do campo definido como <i>input</i>. Para fechar a presente linha, temos o campo <i>value</i> (valor) que representa um valor inicial do campo. Esse valor será modificado com a entrada a ser feita pelo usuário e passado ao <i>script</i> como mencionado agora há pouco.</p>
size e maxlength	<p>As entradas referenciadas nas linhas 25, 26 e 27 trazem, como um novo elemento, a definição do tamanho do campo.</p> <pre><!-- 25 --> <input type="text" name="dia" size="2" maxlength="2" value="dd"> <!-- 26 --> <input type="text" name="mes" size="2" maxlength="2" value="mm"> <!-- 27 --> <input type="text" name="ano" size="4" maxlength="4" value="aaaa"></pre> <p>O parâmetro <i>size</i> define o tamanho do campo e <i>maxlength</i> define o comprimento máximo que o usuário poderá digitar. Neste caso, os valores são iguais, pois data não precisa ter flexibilidade entre o tamanho máximo e o tamanho do campo – diferentemente de, por exemplo, um nome que pode ter um tamanho máximo inferior ao tamanho do campo.</p>
option	<p>A entrada que inicia na linha 35 oferece ao usuário uma possibilidade de escolha dentre as opções presentes nas linhas 36 a 40.</p> <pre><!-- 35 --> <select name="Reg"> <!-- 36 --> <option value="N">Norte</option> <!-- 37 --> <option value="NE">Nordeste</option> <!-- 38 --> <option value="CO">Centro-oeste</option> <!-- 39 --> <option value="SE">Sudeste</option> <!-- 40 --> <option value="S">Sul</option></pre> <p>Você pode notar que, nas linhas 36 a 40 aparece o parâmetro <i>value</i> não para ser impresso na tela mas, sim, o valor a ser passado e testado no <i>script</i>. A informação a ser impressa no formulário, a título de orientação de navegação ao usuário, aparece entre as <i>tags</i> de marcação de início <code><option></code> e marcação de finalização <code></option></code>.</p>
	<p>Por fim, temos, nas linhas 52 e 53, entradas relacionadas a botões.</p> <pre><!-- 52 --> <input type="submit" value="Enviar"> <!-- 53 --> <input type="reset" value="Limpar Formulário"></pre> <p>As ações a serem desempenhadas pelos botões são determinadas pelo seu tipo: <i>submit</i> (submeter/enviar), ou <i>reset</i> (resetar/limpar). Dessa forma, o tipo <i>submit</i> (<code>type=submit</code>), tem</p>

submit e reset o objetivo de enviar os valores de todas as entradas para o *script* referenciado na definição do formulário (linha 9, parâmetro *action*). O tipo *reset* (*type=reset*) realiza a volta ao estado inicial dos campos do formulários, limpando toda a digitação que, porventura, tenha sido realizada. Em ambos os casos, o parâmetro *value* determina o que aparece escrito para o usuário.

VOCÊ QUER LER?



Foram introduzidos novos tipos de entrada na versão 5 do HTML. Para você saber os novos e todos os demais tipos de entradas possíveis de serem usados em seus formulários, recomendamos o artigo “<input>” (MDN WEB DOCS, 2017), disponível em <<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/input>>.

No momento do desenvolvimento de páginas com formulário, deve-se atentar ao fato de que alguns tipos não são cobertos por todos os tipos de navegadores. Desta forma, deve-se fazer, antes, uma pesquisa em relação à compatibilidade de forma a propiciar uma maior flexibilidade em relação ao uso das páginas implementadas. Até o momento, toda a formatação dos componentes e conteúdos a serem exibidos na página foram formatados diretamente com HTML básico. Porém, como deixar a sua página com a aparência melhor? Vamos entender isso, estudando sobre CSS, a seguir.

1.4 Formatação HTML usando CSS

Como já mencionamos, construir páginas usando somente o HTML como elemento de *layout* e *design* não permite que a sua página tenha uma formatação visual mais elaborada. Para resolver essa limitação do HTML podem ser utilizadas as folhas de estilo, ou simplesmente denominadas como CSS (*Cascading Style Sheets* – folhas de estilo em cascata). Sendo assim, a codificação HTML passa a ser somente para definir a estrutura da página, deixando a formatação visual para o CSS (MILETTO; BERTAGNOLLI, 2014). Com o CSS permite-se, também, configurar, de uma só vez, todas as ocorrências de uma *tag* específica HTML.

VOCÊ QUER LER?



Na criação de *sites*, temos que ter atenção à experiência do usuário de forma a poder melhorá-la. Uma das técnicas consiste no chamado “*Layout* responsivo”. Para saber mais a respeito, leia o artigo “*Design* Responsivo na prática 2: do *layout* ao HTML” (GUERRATO, 2014), disponível em <<https://tableless.com.br/design-responsivo-na-pratica-2-layout-ao-html/>>.

Mas, como usar CSS? Na seção a seguir, abordaremos esse assunto de como usar as folhas de estilo em sua página.

1.4.1 Tipos de seletores

Quando se manipula CSS, devemos, antes, explicitar a qual aspecto serão aplicadas as configurações CSS. Ao aspecto selecionado, como por exemplo, mudar o estilo de um cabeçalho, é atribuído o nome *seletor*. Mas, como configurar o seletor? De acordo com Miletto e Bertagnolli (2014), existem três formas de usar CSS: de forma *inline*, internamente e externamente. Clique na seta, para movimentar as abas abaixo e confira.

Inline

Na utilização de CSS inline, deve-se configurar a aparência dentro de cada item aplicando-se o parâmetro `style`, como no exemplo a seguir: `<h2 style="font-family: Times; font-size: 14pt; color:red;">texto</h2>` Na linha de código acima, temos o emprego de CSS sobre o seletor `<h2>`. Sendo assim, a configuração do cabeçalho (configurado para usar a fonte da família "Times", tamanho 14 pontos e na cor vermelha) será aplicada apenas no item "texto", presente na linha de código.

Interna

Mas, existe uma forma de não precisar repetir a configuração a cada utilização de, por exemplo, cabeçalho `h2`? Sim, para isso usaremos a configuração CSS denominada como interna. Antes de mais nada, na forma interna, será necessário habilitar o CSS por meio das tags de marcação de início `<style>` e finalização `</style>`: `<style type="text/CSS">` Entre as marcações de início e finalização, estarão as configurações CSS que você deseja incorporar em sua página. Nessa forma de configuração, a configuração para, por exemplo, `h2`, passaria a ser: `h2 {font-family: Times; font-size: 14pt; color:red;}` Lembrando que a linha de código acima deve estar envolvida pelas tags `<style>` e `</style>`. Junto à definição do `h2`, estarão presentes todos os demais seletores configurados por CSS.

Externa

Convém mencionar que a definição CSS, tanto interna, quanto externa deverá ser realizada dentro do cabeçalho da página, ou seja, entre as tags de marcação `<head>` e `</head>`.

Na terceira forma de configuração CSS, a externa, deve-se especificar o endereço HTML que contém a página de definições CSS. Desta forma, a tag `<style>` aparecerá no seguinte formato: `<link href="local regras CSS.css" rel="stylesheet" type="text/CSS"/>` Na linha de código acima, o arquivo de definições CSS será importado pelos navegadores através do seletor de tags: seletor de classe, seletor de IDs, seletor de atributo, seletores de pseudo-classes e pseudo-elementos (MILETTO, BERTAGNOLLI, 2014).

Para exemplificar a utilização de seletores, vamos abordar os seletores de classe, de IDs e de atributo. Para isso, utilizaremos o código da figura abaixo.

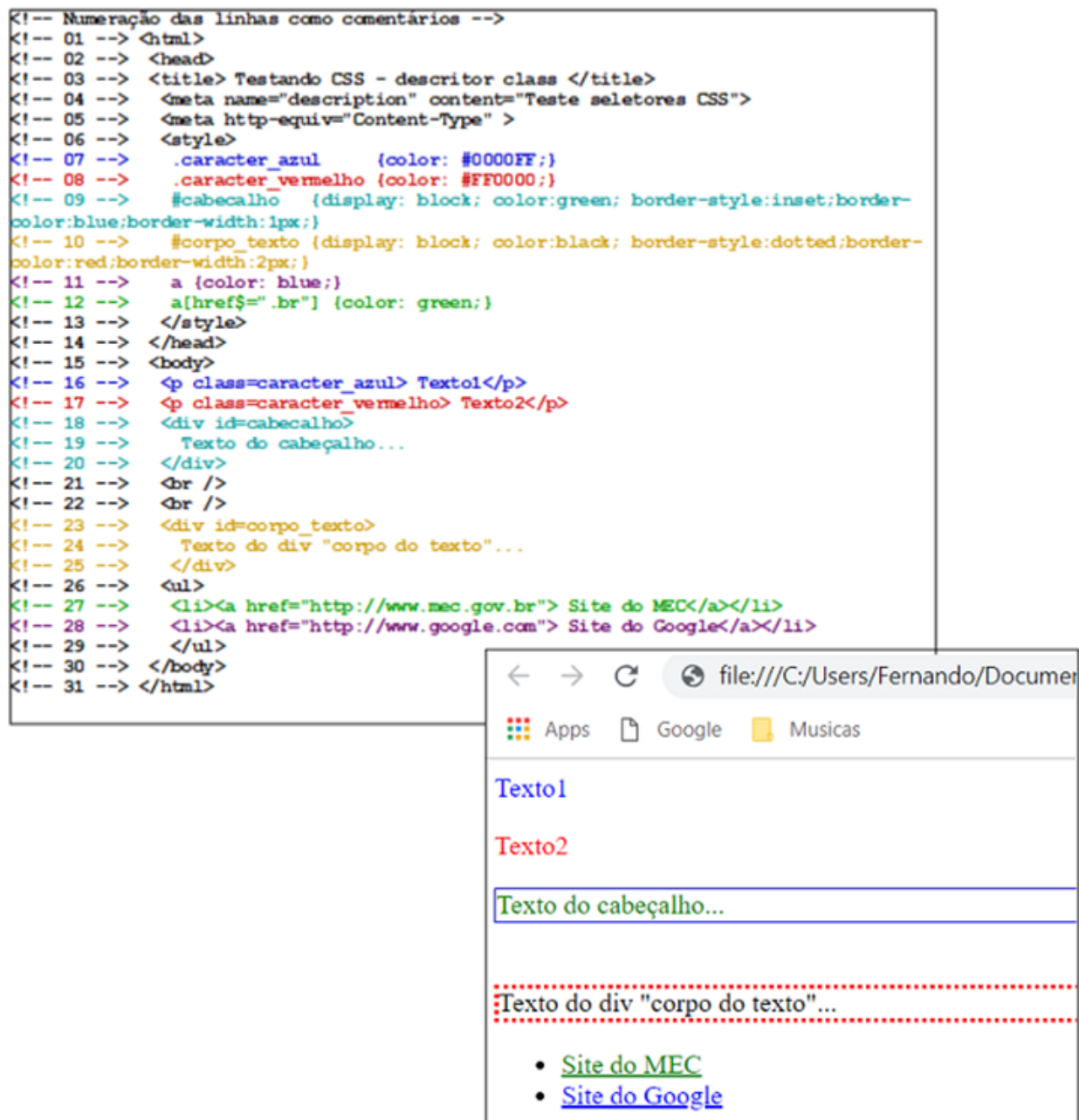


Figura 6 - Exemplo de código HTML para a manipulação de seletores CSS do tipo classe, ID e atributo. A utilização dos seletores se distingue pelo uso de cores distintas.

Fonte: Elaborada pelo autor, 2018.

Na figura acima, encontramos a exemplificação de seletores CSS. Como já mencionado, a definição do estilo CSS encontra-se na seção de cabeçalhos da página – delimitada pelas tags <head> e </head>. Clique nas abas abaixo e compreenda.

-

Seletores de classe

Nas linhas 7 e 8, temos seletores de classe (identificados pelo ponto antes do nome – exemplo: “.caracter_azul”).

```
<!-- 07 --> .caracter_azul {color: #0000FF;}
<!-- 08 --> .caracter_vermelho {color: #FF0000;}
```

Seletores de classe são capazes de modificar o estilo visual de todos os elementos que tiverem, como parâmetro, o nome da classe associado - como ocorre nas linhas 16 e 17.

```
<!-- 16 --> <p class=caracter_azul> Texto1</p>
<!-- 17 --> <p class=caracter_vermelho> Texto2</p>
```

-

Seletores de IDs

Uma ação semelhante ocorre com os seletores de IDs (precedidos por um sinal “#”, como exemplificados nas linhas 9 e 10).

```
<!-- 09 --> #cabecalho {display: block; color:green; border-style:inset;border-
color:blue;border-width:1px;}
<!-- 10 --> #corpo_texto {display: block; color:black; border-style:dotted;border-
color:red;border-width:2px;}
```

Os seletores de IDs são usados quando o item aparece apenas uma vez na página – como é o caso do cabeçalho e corpo da página.

-

Seletores de atributos

Por fim, os seletores de atributos (linhas 11 e 12) configuram o elemento com base na comparação do valor de seus atributos.

```
<!-- 11 --> a {color: blue;}
<!-- 12 --> a[href$=".br"] {color: green;}
```

Os seletores de IDs são usados quando o item aparece apenas uma vez na página – como é o caso do cabeçalho e corpo da página.

VOCÊ QUER LER?



Existem diversos outros seletores CSS. O artigo “Os 30 Seletores CSS Que Você Deve Memorizar” (WAY, 2011) comenta sobre seletores que poderão ser úteis na construção de sua página. O artigo poderá ser acessado pelo *link* <<https://code.tutsplus.com/pt/tutorials/the-30-CSS-selectors-you-must-memorize--net-16048>>.

Para encerrar o assunto sobre seletores CSS Miletto e Bertagnolli (2014) aborda os seletores de pseudoclasse e pseudoelemento. Aqui não vamos entrar em detalhes, teremos apenas uma breve descrição, por meio das tabelas contidas na figura abaixo.

Pseudoclasse	Exemplo	Descrição
:link	a:link{color:#336; text-decoration: none;}	O estilo padrão utilizado para links não visitados é azul e sublinhado. O exemplo altera essas propriedades para outra cor, sem sublinhado.
:visited	a:visited {color:yellow;}	Utilizado para alterar propriedades de links já visitados.
e:hover	a:hover {background-color: red;}	Possibilita alterar a aparência do elemento quando o ponteiro do mouse passa em cima do elemento. Pode ser aplicado a links (como no exemplo) e a outros elementos do HTML.
e:active	a:active{color:black;}	É ativado quando o usuário permanece com o mouse clicado sobre o elemento. Também pode ser aplicado a links e a outros elementos do HTML.
e: focus	a: focus{background-color: red;}	É semelhante ao seletor <code>hover</code> . Enquanto o <code>hover</code> é ativado pelo mouse, o <code>focus</code> é ativado via teclado, geralmente pela tecla TAB.

Pseudoelemento	Exemplo	Descrição
e::first-letter	p::first-letter{color: blue; border:dotted;}	Permite destacar a primeira letra do elemento que está sendo formatado.
e::first-line	p::first-line{color:red; letter-spacing: 6px;}	Possibilita destacar a primeira linha do elemento que está sendo formatado.
e::before e::after	p::before{content: Obs.:}	Permitem a inclusão de conteúdo antes (<i>before</i>) ou depois (<i>after</i>) do conteúdo que já consta na TAG. O conteúdo adicionado deve ser definido dentro do CSS, usando a propriedade <code>content</code> .

Figura 7 - Quadro de resumo sobre os seletores CSS do tipo pseudoclasse e pseudoelemento.

Fonte: Adaptado de MILETTO; BERTAGNOLLI, 2014, p. 78.

Você deve estar se perguntando: existe algo para facilitar a criação de páginas? A resposta é positiva. Existem *frameworks* que facilitarão a criação de suas páginas. Um destes *frameworks* é chamado de Bootstrap. Vamos saber um pouco mais sobre ele a seguir.

1.4.2 Bootstrap

Foi comentado que Bootstrap ([s/d]) é um *framework*. Mas, inicialmente, o que vem a ser um *framework*? *Framework* é um conjunto de métodos que provém soluções comuns para o desenvolvimento de aplicações.

Uma das vantagens no uso do Bootstrap reside no fato de que se pode incorporar as definições CSS pela seguinte linha de código no cabeçalho da página:

```
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css" integrity="sha384-Gn5384xqQ1aoWXA+058RXPxPg6fy4IWvTNh0E263XmFcJlSAwiGgFAW/dAiS6JXm" crossorigin="anonymous">
```

Além de exportar configurações CSS, o Bootstrap também exporta componentes JavaScript, para facilitar a construção de páginas, inclusive permitindo *layout* responsivo.

VOCÊ QUER VER?



Para aprender mais sobre Bootstrap, recomendamos o vídeo “Criando um *site* com Bootstrap {reprograma}” (GERBAUDO, 2016), com dicas práticas para criar um *site* completo e com acessibilidade em *gadgets*. Você pode assistir pelo link: <https://www.youtube.com/watch?v=TC_66K6rkGA>.

Assista ao vídeo e conheça mais sobre os conceitos iniciais envolvidos na criação de páginas, associando-os à própria confecção de páginas baseadas em HTML. Vamos lá?!

https://cdnapisec.kaltura.com/p/1972831/sp/197283100/embedIframeJs/uiconf_id/30443981/partner_id/1972831?iframeembed=true&playerId=kaltura_player_1545316399&entry_id=1_5u8bsvka

Existem diversas interfaces que implementam o *framework* Bootstrap. Dentre as quais, podemos destacar: Bootstrap UI, Bootstrap Studio, Bootsnipp, LayoutIT! e Shards. Além do Bootstrap, você poderá contar, também, com diversos outros *frameworks* – dentre os quais podemos citar: Foundation, Materialize, Semantic UI, Material UI e Pure. Cada um com as suas particularidades e facilidades.

Todos esses *frameworks* são definidos como *frontend* uma vez que atuam na definição da interface a ser manipulada pelo usuário. Por sua vez, os *frameworks backend* tem por objetivo atuar junto às máquinas servidoras, responsáveis por providenciar as requisições realizadas pelo usuário por intermédio da camada *frontend* (interface). Como exemplos de *frameworks backend* temos: Django, Ruby on Rails, Flask, Phoenix, Express.js, Laravel e Pyramid.

Síntese

Chegamos ao fim de nosso primeiro capítulo. Pudemos tratar alguns aspectos iniciais para o desenvolvimento de páginas baseadas em HTML, adicionando configurações de *layout* CSS. Lembre-se que, para desenvolver uma página, você deve iniciar com o seu projeto, incluindo elementos de usabilidade e acessibilidade. Somente depois de tudo definido é que se parte para a implementação.

Como mencionado aqui, a implementação pode contar com a ajuda de *frameworks* e ambientes de desenvolvimento para que o processo possa ser desenvolvido de forma mais eficiente e organizada.

Com os pontos abordados neste capítulo, você já tem uma boa base para desenvolver suas primeiras páginas, experimentando os elementos aqui abordados e buscando diversos outros. O desenvolvimento de páginas envolve muitos recursos já disponíveis e a cada dia, surgem mais, em uma velocidade muito rápida, por isso, é importante se atualizar constantemente.

Neste capítulo, você teve a oportunidade de:

- conhecer os conceitos de HTML;
- identificar e aplicar as *tags* HTML básicas;
- analisar e aplicar a estruturação de páginas;
- compor e experimentar funcionalidades e estruturas inerentes a tabelas;
- estruturar e implementar tabelas mescladas;
- planejar e implementar formulários em páginas HTML;
- planejar e propor alterações de *layout* pelo uso de CSS;
- conhecer *frameworks* para o desenvolvimento de páginas.

Bibliografia

- ARRIGONI, R. **Código Tag – O que é meta tag?** Portal Linha de Código, publicado em 09/10/2012. Disponível em <<http://www.linhadecodigo.com.br/artigo/3595/codigo-tag-o-que-e-meta-tag.aspx>>. Acesso em 10/10/2018.
- ASSUMPÇÃO, D. J. F.; VILLEGAS, G. M. L. G. C. **Processo de criação em websites: um estudo de caso do site IESAM.** Dito Efeito. Curitiba, vol. 4, n. 4, 2013. Disponível em <<https://periodicos.utfpr.edu.br/de/article/download/2129/2028>>. Acesso em: 23/10/2018.
- BOOTSTRAP. **Bootstrap documentation.** [s/d]. Disponível em <<https://getBootstrap.com/docs/4.0/getting-started/introduction/>>. Acesso em: 23/10/2018.
- FÁBIO, A. C. **Tim Berners-Lee, o criador da internet global. E suas críticas à rede hoje.** Portal Nexa, publicado em 13/03/2018. Disponível em: <<https://www.nexojornal.com.br/expresso/2018/03/13/Tim-Berners-Lee-o-criador-da-internet-global.-E-suas-cr%C3%ADticas-%C3%A0-rede-hoje>>. Acesso em: 23/10/2018.
- GERBAUDO, R. **Criando um site com Bootstrap {reprograma}.** Canal Ricardo Gerbaudo, YouTube, publicado em 23/10/2016. Disponível em <https://www.youtube.com/watch?v=TC_66K6rkGA>. Acesso em: 23/10/2018.
- GUERRATO, D. **Design responsivo na prática 2: do Layout ao HTML.** Publicado em 28/04/2014. Disponível em: <<https://tableless.com.br/design-responsivo-na-pratica-2-layout-ao-HTML/>>. Acesso em: 23/10/2018.
- JOÃO, B. N. **Informática Aplicada.** São Paulo, Pearson Education do Brasil, 2014. [Recurso eletrônico, Biblioteca Virtual Universitária]
- LAUBE, K. P. **Entendendo o CGI, FastCGI e WSGI.** Portal Klauslaube, publicado em 02/11/2012. Disponível em <<https://klauslaube.com.br/2012/11/02/entendendo-o-cgi-fastcgi-e-wsgi.HTML>>. Acesso em: 23/10/2018.
- LEMAY, L.; COLBURN, R.; TYLER, D. **Aprenda a criar páginas Web com HTML e XHTML em 21 dias.** São Paulo: Pearson Education do Brasil, 2002. [Recurso eletrônico, Biblioteca Virtual Universitária]
- MILETTO, E. M.; BERTAGNOLLI, S. C. **Desenvolvimento de software II: introdução ao desenvolvimento web com HTML, CSS, javascript e php.** Porto Alegre, Bookman, 2014. [recurso online, Minha Biblioteca]
- MDN WEB DOCS. **<input>.** Portal MDN web docs, publicado em 14/10/2017. Disponível em: <<https://developer.mozilla.org/pt-BR/docs/Web/HTML/Element/input>>. Acesso em: 23/10/2018.
- PAGANI, T. **O que é usabilidade?** Portal Tableless, publicado em 22/08/2011. Disponível em: <<https://tableless.com.br/o-que-e-usabilidade/>>. Acesso em: 23/10/2018.
- QUEIROZ, M. A.; PORTA, G. **Criação de tabelas de dados acessíveis.** Portal Acessibilidade Legal, publicado em 22/12/2009. Disponível em <<http://www.acessibilidadelegal.com/13-tabelas-acessiveis.php>>. Acesso em: 23/10/2018.
- VICTORIO, J. **Requisições GET e POST – principais diferenças.** Publicado em 24/07/2016. Disponível em <<http://tecnologiaeinovacao.com.br/blog/2016/07/24/requisicoes-get-e-post-principais-diferencas/>>. Acesso em: 23/10/2018.
- W3SCHOOLS. **HTML 5 Tutorial.** [s/d]. Disponível em: <<https://www.w3schools.com/HTML/>>. Acesso em: 23/10/2018.

WAY, J. **Os 30 seletores CSS que você deve memorizar**. Portal Envato Tuts+, publicado em 09/06/2011.
Disponível em: <<https://code.tutsplus.com/pt/tutorials/the-30-CSS-selectors-you-must-memorize--net-16048>>.
Acesso em: 23/10/2018.