

RESPONSIVE WEB DEVELOPMENT

HTML - O INÍCIO

LISTA DE FIGURAS

Figura 2.1 – Padrão de codificação HTML, derivado de SGML	6
Figura 2.2 – Tag <!DOCTYPE> no padrão HTML5	7
Figura 2.3 – Tag <!DOCTYPE> no padrão XHTML 1.0	7
Figura 2.4 – Tag <html> no padrão XHTML 1.0	8
Figura 2.5 – Tag <html> no padrão HTML5	8
Figura 2.6 – Tag <head>	10
Figura 2.7 – Tag <title>	11
Figura 2.8 – Aba exibindo o título do documento	11
Figura 2.9 – Tag <link> vinculando arquivos CSS no padrão HTML5	12
Figura 2.10 – Tag <link> vinculando arquivos CSS no padrão XHTML 1.0	12
Figura 2.11 – Tag <link> vinculando arquivo de ícone no padrão HTML5	13
Figura 2.12 – Documento HTML com ícone, um dos usados da tag <link>	13
Figura 2.13 – Tag <meta> sendo utilizada para definir o padrão de codificação de caracteres	14
Figura 2.14 – Tag <meta> sendo utilizada para recarregar o documento HTML depois de 5	15
Figura 2.15 – Tag <meta> sendo utilizada para definir as palavras-chave de um documento	15
Figura 2.16 – Tag <style>	16
Figura 2.17 – Tag <style> em funcionamento	16
Figura 2.18 – Tag <script> com código JavaScript embutido no HTML	17
Figura 2.19 – Tag <script> com referência à um arquivo externo JavaScript ..	17
Figura 2.20 - Tag <body>	18
Figura 2.21 – Tag <p> aplicada ao HTML	19
Figura 2.22 – Tag <p> em funcionamento	19
Figura 2.23 – Tag aplicada ao HTML	19
Figura 2.24 – Tag em funcionamento	20
Figura 2.25 – Uso das tags <h1>, <h2> e <h3> aplicada ao HTML	21
Figura 2.26 – Tags <h1>, <h2> e <h3> em funcionamento	21
Figura 2.27 – Tag aplicada ao HTML	23
Figura 2.28 – Tag em dois momentos: Sem localizar a imagem origem e com a mesma localizada	23
Figura 2.29 - Tag <div> aplicada ao HTML	24
Figura 2.30 – Tag <div> em funcionamento	24
Figura 2.31 – Exemplo de uso das novas tags semânticas	27
Figura 2.32 – Tag <a> implementada em HTML	28
Figura 2.33 – Tag <a> em funcionamento	28
Figura 2.34 – Tags <a> e criando um hyperlink em uma imagem	28
Figura 2.35 – Hyperlink aplicado em uma imagem	29
Figura 2.36 – Tags e criando uma lista ordenada	29
Figura 2.37 – Lista ordenada	30
Figura 2.38 – Tags e criando uma lista não ordenada	30
Figura 2.39 – Lista não ordenada	30
Figura 2.40 – O arquivo iframe_pagina1.html	31
Figura 2.41 – O arquivo iframe_pagina2.html	31

Figura 2.42 – O arquivo <code>iframe.html</code>	32
Figura 2.43 – <code>iframe</code> em dois momentos: Na carga da página e ao clicar no hyperlink disponível	32
Figura 2.44 – Tag <code><table></code> sendo utilizada	34
Figura 2.45 – Tabela criada	35
Figura 2.46 – Tags <code><thead></code> , <code><tbody></code> e <code><tfoot></code> sendo utilizadas	35
Figura 2.47 – Tags <code><thead></code> , <code><tbody></code> e <code><tfoot></code> sem suas <i>tags</i> de finalização	36
Figura 2.48 – Linhas de tabela sendo definidas com a <i>tag</i> <code><tr></code>	37
Figura 2.49 – Exemplo de tabela usando <code><td></code> e <code><th></code>	39
Figura 2.50 – Tabela completa	39
Figura 2.51 – Mudando as dimensões de uma célula (e coluna)	40
Figura 2.52 – Dimensionando a primeira célula (e coluna) em 50% do tamanho total da tabela	41
Figura 2.53 – Mesclando células horizontalmente	42
Figura 2.54 – Mesclando células horizontalmente	43
Figura 2.55 – Mesclando células verticalmente	43
Figura 2.56 – Mesclando células verticalmente	44
Figura 2.57 – Exemplo de aplicação da tag <code><form></code> em HTML	46
Figura 2.58 – Exemplo de uso da tag <code><label></code> em HTML	46
Figura 2.59 – Rótulo para campos de formulário	47
Figura 2.60 – Exemplo de formulário usando <code><input></code> com <code>value</code> e <code>disabled</code> ...	50
Figura 2.61 – Exemplo de formulário usando <code><input></code> com <code>value</code> e <code>disabled</code> ...	50
Figura 2.62 – Exemplo de formulário usando <code><input></code> com <code>checked</code>	51
Figura 2.63 – Exemplo de formulário usando <code><input></code> com <code>checked</code>	51
Figura 2.64 – Exemplo de formulário usando <code><input></code>	54
Figura 2.65 – Exemplo de formulário usando <code><input></code>	54
Figura 2.66 – Exemplo de formulário usando <code><select></code> e <code><option></code> com opção previamente selecionada	55
Figura 2.67 – Exemplo de formulário usando <code><select></code> e <code><option></code> com opção previamente selecionada	56
Figura 2.68 – Exemplo de formulário usando <code><select></code> , <code><option></code> e <code><optgroup></code> com seleção múltipla	57
Figura 2.69 – Exemplo de formulário usando <code><select></code> , <code><option></code> e <code><optgroup></code> com seleção múltipla	58
Figura 2.70 – Exemplo de formulário usando <code><textarea></code>	59
Figura 2.71 – Exemplo de formulário usando <code><textarea></code>	59
Figura 2.72 – Exemplo de formulário usando <code><fieldset></code> e <code><legend></code>	60
Figura 2.73 – Exemplo de formulário usando <code><fieldset></code> e <code><legend></code>	60
Figura 2.74 – Exemplo de formulário usando <code><datalist></code>	62
Figura 2.75 – Exemplo de formulário usando <code><datalist></code> em dois momentos: usando como caixa de seleção e digitando uma opção não encontrada na lista	62
Figura 2.76 – Exemplo de formulário usando <code><progress></code>	63
Figura 2.77 – Exemplo de formulário usando <code><progress></code>	63

SUMÁRIO

2 HTML - O INÍCIO.....	5
2.1 Considerações sobre os diferentes padrões	5
2.1.1 Padrão de codificação	6
2.2 Tags essenciais.....	7
2.2.1 Tag <!DOCTYPE>	7
2.2.2 Tag <html>	8
2.2.3 Atributos Globais	9
2.2.4 Tag <head>	10
2.2.5 Tag <title>	10
2.2.6 Tag <link>.....	11
2.2.7 Tag <meta>	14
2.2.8 Tag <style>	15
2.2.9 Tag <script>	16
2.2.10 Tag <body>	18
2.3 Tags básicas	18
2.3.1 Tag <p>	18
2.3.2 Tag 	19
2.3.3 Tags <h1>, <h2>, <h3> ... até <h6>	20
2.3.4 Tag 	22
2.3.5 Tag <div>	24
2.3.6 Substitutivos do <div> para melhorar a semântica	24
2.3.7 Tag <a>	27
2.3.8 Tags , e 	29
2.3.9 Tag <iframe>	31
2.3.10 Tags <frame> e <frameset>: Por que não usar?.....	33
2.3.11 Tags de marcação a se esquecer	33
2.4 Tabulação de dados: criando tabelas.....	33
2.4.1 Tag <table>	34
2.4.2 Tags <thead>, <tbody> e <tfoot>	35
2.4.3 Tag <tr>	36
2.4.4 Tags <td> e <th>	37
2.4.5 Dimensionando células	39
2.4.6 Mesclando colunas.....	41
2.5 Criando formulários	44
2.5.1 Tag <form>.....	44
2.5.2 Tag <label>	46
2.5.3 Tag <input>	47
2.5.4 Tags <select>, <option> e <optgroup>.....	54
2.5.5 Tag <textarea>	58
2.5.6 Tags <fieldset> e <legend>	59
2.5.7 Tag <datalist>.....	61
2.5.8 Tag <progress>	62
2.6 Considerações finais	63
REFERÊNCIAS	64

2 HTML - O INÍCIO

A linguagem de marcação de hipertexto HTML foi criada com o intuito de diagramar documentos para o serviço de World Wide Web (WWW). Em seus primórdios, foi concebida pelo físico britânico Tim Berners-Lee para reproduzir documentos de pesquisa e compartilhá-los com facilidade. O padrão logo foi adotado pelas universidades que começavam a fazer parte da Internet, na década de 1990, na reprodução e compartilhamento de teses acadêmicas.

A WWW, no entanto, evoluiu muito nas últimas décadas: ela é usada de forma comercial, divulgando e vendendo produtos e serviços de empresas, estreitando os relacionamentos e comunicação em redes sociais e no entretenimento, veiculando áudio, vídeo e jogos. Uma boa linguagem, seja ela de programação ou marcação, é realmente boa quando permanece viva, ou seja, é constantemente atualizada para se adequar à realidade atual. Felizmente o HTML é uma destas linguagens.

2.1 Considerações sobre os diferentes padrões

A evolução mencionada gerou várias especificações diferentes. Muitas páginas na Internet estão escritas no padrão HTML 4.01, publicado em 24 de dezembro de 1999. Uma reformulação do HTML 4 foi lançada quase um mês depois, conhecida como Extensible HyperText Markup Language, o XML 1.0, reformulado em 1º de agosto de 2002. E, finalmente, outras páginas utilizam a revisão, o XML 1.1, cuja segunda versão foi liberada em 16 de agosto de 2006.

Concentraremos nossos estudos no último padrão vigente, o HTML 5, que tem sido trabalhado intensamente nos últimos anos e foi considerado “recomendado para uso” pela W3C, em 28 de outubro de 2014. Explico o porquê: o padrão foi rapidamente adotado por todos os fabricantes de navegadores do mercado, incluindo os usados em dispositivos portáteis.

De acordo com o site W3Counter em números de dezembro de 2015, aproximadamente 70% utilizam versões de navegadores extremamente atualizados (Google Chrome em versões 38, 46 e 47, Mozilla Firefox 42 e 43,

Internet Explorer 11, Safari versões 8 e 9, entre outros). O padrão HTML5 seria um grande problema para navegadores Internet Explorer nas versões 8 para baixo. Pois bem, sendo a mesma fonte, apenas 1,9% dos visitantes estão utilizando IE nas versões 6, 7 e 8.

Sendo assim, utilize o padrão HTML 5 em novos documentos HTML. Exceções devem ser feitas para um ambiente Intranet, cuja empresa possui um parque tecnológico extremamente defasado; ou aplicações web desenvolvidas com foco em especificidades do IE 6, incompatíveis com as versões posteriores, forçando gestores de TI a proibir atualizações de navegadores web (Obrigado, Microsoft!). Nestes casos, recomenda-se o padrão XML 1.0.

2.1.1 Padrão de codificação

A linguagem HTML deriva da metalinguagem Standard Generalized Markup Language (SGML), inventada pela IBM da década de 1960. As sintaxes de marcação são informadas entre "<" (sinal de menor) e ">" (sinal de maior), e serão chamadas a partir de agora de tags. Estas, por sua vez, possuem uma hierarquia, ou seja, existem "subtags" que só devem ser posicionadas dentro de uma determina tag "mãe". Por esta razão, a maioria das tags HTML são abertas e fechadas (o fechamento usa o símbolo "/", barra).

Veja o exemplo:

```
<tag>
  <subtag  atributo="valor">Informação  cuja  marcação
atuará</subtag>
</tag>
```

Figura 2.1 – Padrão de codificação HTML, derivado de SGML
Fonte: Elaborado pelo autor (2016)

As tags deste exemplo são fictícias, mas dão uma boa ideia do que virá pela frente. Tags possuem atributos que possibilitam a parametrização do efeito gerado pela tag, e segue o padrão exibido no exemplo: atributo="valor".

2.2 Tags essenciais

Vamos começar com as tags essenciais, ou seja, aquelas que todo documento HTML precisa possuir.

2.2.1 Tag <!DOCTYPE>

O Document Type Definition (DTD) ou a tag <!DOCTYPE> não é exatamente parte do padrão HTML, mas é muito importante. Conforme dito anteriormente, existem vários padrões HTML. Pois bem, como o navegador do usuário sabe em qual destes padrões do documento HTML foi escrito? Bem, ele não sabe, ele “adivinha” qual é, e nem sempre acerta.

Trata-se da primeira tag de um documento, colocada antes mesmo da <html>. Existem dezenas de doctypes diferentes que ao serem interpretadas pelo navegador não só determinam o padrão a ser seguido, como ativam modos de renderização diferentes: standards mode é seu modo peculiar de renderização, quirks mode é utilizado para páginas antigas ou o modo intermediário quase padronizado conhecido como almost standards mode (sei que parece piada, mas não é).

Para o padrão HTML5, a tag foi devidamente encurtada, não exigindo a especificação DTD determinada. A linha de código fica assim:

```
<!DOCTYPE html>
```

Figura 2.2 – Tag <!DOCTYPE> no padrão HTML5
Fonte: Elaborado pelo autor (2016)

Caso o documento for escrito em XHTML 1.0, a tag <!DOCTYPE> fica assim:

```
<!DOCTYPE html  
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"  
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-  
strict.dtd">
```

Figura 2.3 – Tag <!DOCTYPE> no padrão XHTML 1.0
Fonte: Elaborado pelo autor (2016)

2.2.2 Tag <html>

A tag <html> determina onde começa e termina o documento HTML. Todas as outras tags devem ser contidas dentro dela. Em XHTML 1.0 a declaração era:

```
<!DOCTYPE html
    PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    lang="pt-br" xml:lang="pt-br">
</html>
```

Figura 2.4 – Tag <html> no padrão XHTML 1.0
Fonte: Elaborado pelo autor (2016)

No padrão HTML5, os atributos “xmlns” e “xml:lang” não são mais necessários, e a linha pode ser reduzida para:

```
<!DOCTYPE html>
<html lang="pt-br">
</html>
```

Figura 2.5 – Tag <html> no padrão HTML5
Fonte: Elaborado pelo autor (2016)

O atributo lang é global, ou seja, pode ser utilizado em qualquer *tag* da linguagem HTML. Entretanto, é mais comum que seja informada em <html>, determinando assim o idioma do documento todo.

A importância é vital: Como acontece com <!DOCTYPE>, se o idioma não for indicado aqui, o navegador tenta adivinhar qual é, podendo falhar miseravelmente. Além disso, mecanismos de busca se beneficiam desta indicação de idioma, tornando-se assim mais eficazes. Algoritmos do Google dão preferência a buscas orgânicas para documentos cujo **lang** é utilizado, ou seja, é absolutamente obrigatório. Falaremos muito aqui sobre estas boas práticas relacionadas à SEO (*Search Engine Optimization*).

2.2.3 Atributos Globais

Conforme dito, atributos globais são aqueles que podem ser aplicados em qualquer elemento/tag HTML. A seguir, relacionaremos os principais:

- **accesskey:** com ele, é possível determinar um teclado de atalho para o elemento; ao digitar usando a tecla aqui informada, o elemento HTML em questão ganha foco.
- **class:** este atributo está relacionado com o CSS, que falaremos adiante. Deve ser usado para informar a classe que será utilizada para estilizar o elemento em questão. (Exemplos no capítulo de CSS).
- **id:** utilizado para informar ou identificar um único do elemento. Como o nome já indica, ele deve ser único, ou seja, nenhum outro elemento poderá ter o mesmo id. Este pode ser uma palavra, uma palavra seguida de número, segue basicamente as mesmas regras de batismo a que estamos acostumados ao declarar variáveis em linguagens de programação. Será muito útil quando tratarmos de JavaScript (em um capítulo posterior).
- **lang:** determina o idioma utilizado dentro do elemento HTML.
- **style:** pode ser utilizado para aplicar estilos no padrão CSS diretamente no elemento. Sendo assim, o valor dentro deste atributo é linguagem CSS pura. Mais uma dica de SEO aqui: **style** não é a solução ideal, prefira usar o atributo **class**.
- **tabindex:** por questões de rapidez (ou pela falta do mouse), alguns usuários gostam de navegar entre os elementos HTML usando a tecla TAB do teclado; aliás, esta forma de navegação é muito utilizada por deficientes visuais. Conforme a tecla TAB é acionada, um novo elemento do HTML ganha foco. Pois bem: nem sempre a ordem estabelecida é a mais adequada. Você pode modificar isso atribuindo números (começando em 0, sendo este o primeiro elemento do documento a ganhar foco), assim, a navegação por TAB pode ser inteiramente personalizada.

- **title:** representa informações do elemento, como uma dica (ou *tooltip*). No caso de um hiperlink, pode ser o título ou uma descrição do destino; no caso de uma imagem, pode ser seu crédito ou uma descrição da imagem (essencial para leitores de tela de deficientes visuais); no caso de um parágrafo, pode ser uma nota de rodapé ou comentário sobre o texto.

Não se preocupe, veremos adiante exemplos utilizando os atributos globais aqui descritos.

2.2.4 Tag <head>

Determina o cabeçalho de um documento HTML. Serve, na verdade, como um container de outras tags importantes para o documento, mas não fazem parte do corpo deste propriamente dito:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
  </head>
</html>
```

Figura 2.6 – Tag <head>
Fonte: Elaborado pelo autor (2016)

2.2.5 Tag <title>

Utilizada para informar o título do documento HTML. Como <title> faz parte do cabeçalho (<head>), este título não é renderizado em tela – somente as tags que fazem parte do corpo do documento o serão. Entretanto, não é por isso que não é importante: trata-se do título que será utilizado pelos mecanismos de buscas, topo e abas do seu navegador.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de Título</title>
  </head>
</html>
```

Figura 2.7 – Tag <title>
Fonte: Elaborado pelo autor (2016)

Este código-fonte produz o seguinte resultado:

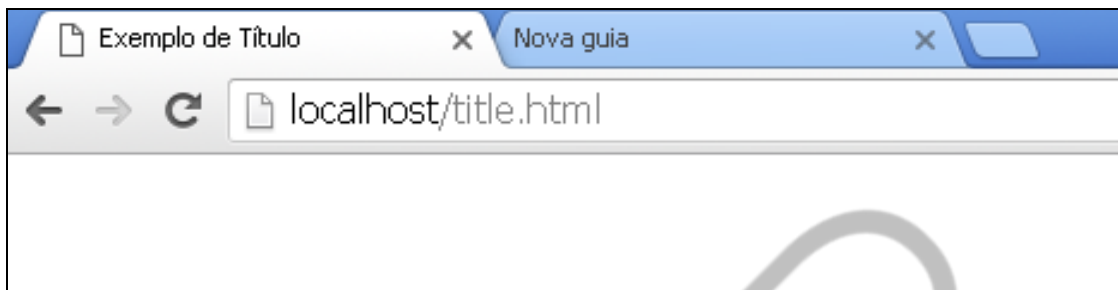


Figura 2.8 – Aba exibindo o título do documento
Fonte: FIAP (2016)

Ufa! Finalmente consegui lhe mostrar algo, não é mesmo? Calma, temos muito pela frente e muitas coisas interessantes para ver.

2.2.6 Tag <link>

<link> não é a tag dos tradicionais hyperlinks clicáveis da Internet: que faz isso é a tag <a>, que veremos mais adiante. Lembre-se: nem começamos a abordar o corpo do documento, esta tag é filha de <head>.

Para que serve <link> então? Vincular documentos. Uma página muitas vezes é composta de vários arquivos: mantemos o HTML, CSS e JavaScript em arquivos separados (esta é a boa prática e, portanto, prática SEO).

Ela é utilizada comumente para vincular os arquivos CSS necessários. Veja o exemplo abaixo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de Título</title>
    <link rel="stylesheet" type="text/css"
href="css/principal.css" title="Estilos principais">
    <link rel="stylesheet" type="text/css"
href="css/titulos.css" title="Titulos">
  </head>
</html>
```

Figura 2.9 – Tag <link> vinculando arquivos CSS no padrão HTML5.

Fonte: Elaborado pelo autor (2016)

Interessante notar que a tag <link> não possui uma tag de fechamento (como </link>). Isso acontece porque ela não está marcando um trecho de texto, não existe conteúdo a ser contido.

Pelo padrão HTML5, a tag é simplesmente aberta e não é fechada, como no exemplo acima. Pelo padrão XHTML, a tag abre e fecha em si mesmo, como pode ser visto no exemplo a seguir:

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
lang="pt-br" xml:lang="pt-br">
  <head>
    <title>Exemplo de Título</title>
    <link rel="stylesheet" type="text/css"
href="css/principal.css" title="Estilos principais" />
    <link rel="stylesheet" type="text/css"
href="css/titulos.css" title="Titulos" />
  </head>
</html>
```

Figura 2.10 – Tag <link> vinculando arquivos CSS no padrão XHTML 1.0

Fonte: Elaborado pelo autor (2016)

Falemos agora dos principais atributos específicos de <link>:

- **href:** abreviação de *hyperlink reference*, trata-se do endereço do arquivo que está sendo vinculado. O endereço pode ser absoluto (caso estejamos acessando um arquivo externo, como `http://www.outrosite.com.br/css/estilos.css`) ou relativo. Um caminho relativo toma como referência o diretório em que o documento HTML está como base. Por exemplo, `href="arquivo.css"` significa que o arquivo CSS está no mesmo diretório que o documento HTML que lhe faz referência. Em nosso exemplo, os dois arquivos estão em uma subpasta chamada "css" (boa prática!).

- **rel:** qual o relacionamento entre os documentos vinculados? Existem várias possibilidades: posso estar vinculado um *stylesheet* (folha de estilo, no caso, CSS) ou um *icon* (ícone), por exemplo. Vários outros valores foram criados para a versão HMTML5 (vide o material/curso específico de HTML5).
- **type:** qual o tipo MIME de documento vinculado? No caso de CSS, o valor deve ser “text/css”.

Que tal um exemplo vinculado a um arquivo para ícone? Embora os navegadores modernos aceitem ícones de imagens em formato PNG, usaremos o padrão, que são ícones no formato ICO:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de Título</title>
    <!-- IE, sempre você... -->
    <link rel="shortcut icon" type="image/x-icon"
href="imagens/html5.ico" />
    <!-- todos os outros navegadores -->
    <link rel="icon" type="image/x-icon"
href="imagens/html5.ico" />
  </head>
</html>
```

Figura 2.11 – Tag <link> vinculando arquivo de ícone no padrão HTML5

Fonte: Elaborado pelo autor (2016)

Esta é uma das ocasiões em que o Internet Explorer diverge do padrão: ele espera que o rel informado seja “shortcut icon”, enquanto todos os outros navegadores esperam o valor “icon”.

A propósito, isso é uma tag de <!-- Comentário -->.



Figura 2.12 – Documento HTML com ícone, um dos usados da tag <link>

Fonte: FIAP (2016)

2.2.7 Tag <meta>

Este elemento é utilizado para informar os metadados do documento HTML. Trata-se de uma tag que também não possui fechamento (</meta>) assim como <link>. Seu uso tem várias aplicações:

- **Padrão de codificação de caracteres:** existem dezenas de padrões de codificação de caracteres, pois temos diversos alfabetos e padrões diferentes de escrita no mundo todo. O padrão utilizado no Brasil é o “iso-8859-1”, conhecido em outros meios como “latin1”.

Existe, no entanto, um padrão unificado, conhecido como “utf-8”, que tem sido amplamente aceito pelos principais portais brasileiros. Recomendo o uso deste padrão, mas tenha o cuidado de salvar o código-fonte exatamente no mesmo padrão em sua suíte de desenvolvimento (sim, arquivos de texto puro também possuem os mesmos padrões de codificação de caracteres).

Não usar a tag <meta> para informar o padrão de codificação fará com que o navegador procure adivinhá-lo e, se tem uma coisa com o potencial de dar realmente errado, é isso. Informá-lo é essencial e faz parte das práticas SEO.

O atributo charset deve ser utilizado para este fim, como pode ser visto no exemplo a seguir:

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

Figura 2.13 – Tag <meta> sendo utilizada para definir o padrão de codificação de caracteres
Fonte: Elaborado pelo autor (2016)

- **Recarregar a página html:** usando os atributos http-equiv e content, podemos definir a rotina de recarga automática da página. O valor de http-equiv será "Refresh" e o content armazena o

número de segundos para a recarga. O exemplo a seguir recarrega a página após 5 minutos:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo meta </title>
    <meta http-equiv="Refresh" content="300">
  </head>
</html>
```

Figura 2.14 – Tag <meta> sendo utilizada para recarregar o documento HTML depois de 5
Fonte: Elaborado pelo autor (2016)

- **Palavras-chave do documento:** com os atributos name e content, podemos utilizar <meta> para armazenar as palavras-chave do conteúdo. Informá-las é importante do ponto de vista SEO, mas é interessante ressaltar que esta tag é apenas um dos fatores analisado pelos motores de busca.

Veja o exemplo a seguir:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo meta</title>
    <meta name="keywords" content="fiap, exemplo, html">
  </head>
</html>
```

Figura 2.15 – Tag <meta> sendo utilizada para definir as palavras-chave de um documento
Fonte: Elaborado pelo autor (2016)

2.2.8 Tag <style>

Permite aplicar estilos CSS diretamente no documento. A boa prática é, no entanto, criar um arquivo .css separado e vinculá-lo com a tag <meta>.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de Título</title>
    <meta name="keywords" content="fiap, exemplo, html">
    <style>
      body { background-color: black; }
    </style>
  </head>
  <body>
  </body>
</html>
```

Figura 2.16 – Tag <style>
Fonte: Elaborado pelo autor (2016)

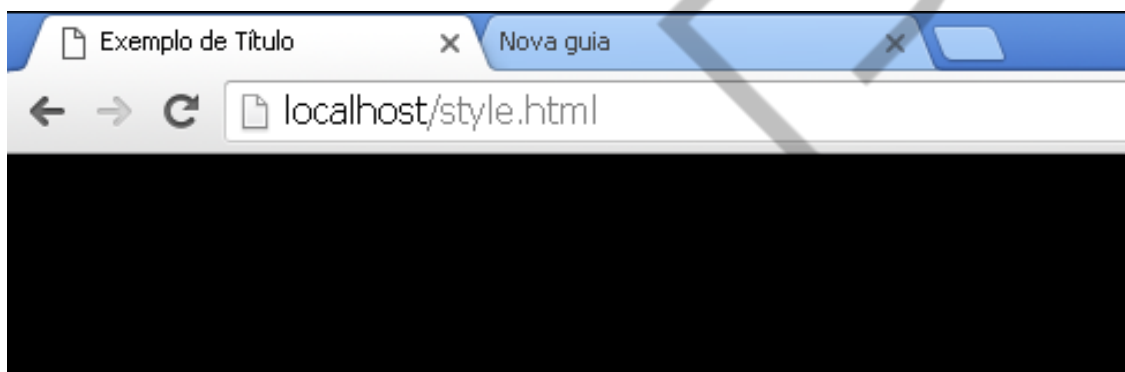


Figura 2.17 – Tag <style> em funcionamento
Fonte: FIAP (2016)

2.2.9 Tag <script>

Utilizado para inserir comandos de linguagem do tipo script que seja interpretado pelo navegador (ou seja, do lado cliente). Resumidamente, JavaScript.

Os comandos da linguagem ficam entre as tags <script>, definindo a linguagem (“text/javascript”) utilizando o atributo “type”. Isso era necessário, pois no passado exibiam outras opções de linguagem do tipo script, como o VBScript.

Veja um exemplo:


```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo JavaScript</title>
  </head>
  <body>
  </body>
</html>
<script type="text/javascript">
  window.alert("Olá, mundo!");
</script>
```

Figura 2.18 – Tag <script> com código JavaScript embutido no HTML
Fonte: Elaborado pelo autor (2016)

Entretanto, essa não é a melhor abordagem, pois os códigos da linguagem JavaScript ficam misturados com a linguagem de marcação HTML. Além de dificultar a manutenção, esta prática não possibilita o reaproveitamento de código, compartilhando as instruções JavaScript com outros documentos HTML (aliás, este raciocínio é aplicável ao caso do CSS).

A boa prática é separá-los, criando um arquivo de extensão .js a parte e vinculá-lo ao documento HTML, utilizando o atributo “src” (source):

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo JavaScript</title>
    <script type="text/javascript"
src="arquivo.js"></script>
  </head>
  <body>
  </body>
</html>
```

Figura 2.19 – Tag <script> com referência à um arquivo externo JavaScript
Fonte: Elaborado pelo autor (2016)

Falaremos bastante sobre o assunto em um capítulo posterior.

2.2.10 Tag <body>

Determina o corpo de um documento HTML. Todas as tags e a partir daqui ficarão contidas dentro deste elemento.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de body</title>
    <meta charset="utf-8">
  </head>
  <body>
  </body>
</html>
```

Figura 2.20 - Tag <body>
Elaborado pelo autor (2016)

2.3 Tags básicas

Seguimos agora para as tags básicas, abordando as tags de agrupamento e marcação de texto.

2.3.1 Tag <p>

Utilizada para definir o início e o fim de um parágrafo. Repare que os navegadores atribuem automaticamente uma margem entre um parágrafo e outro, que pode ser eliminada ou até mesmo aumentada usando CSS.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de Parágrafo</title>
    <meta charset="utf-8">
  </head>
  <body>
    <p>Primeiro parágrafo</p>
    <p>Segundo parágrafo</p>
  </body>
</html>
```

Figura 2.21 – Tag <p> aplicada ao HTML
Fonte: Elaborado pelo autor (2016)

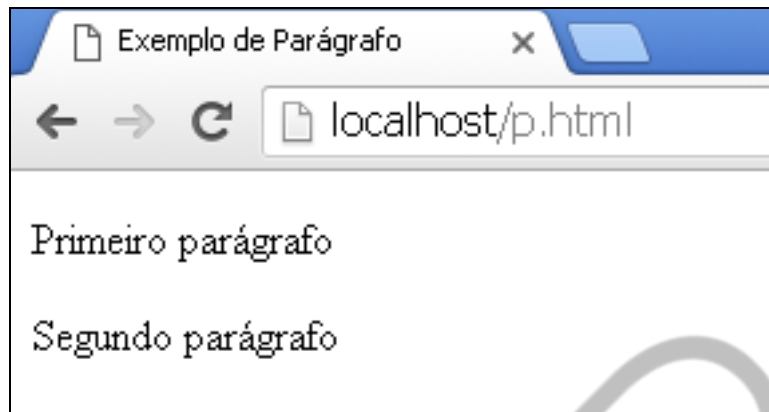


Figura 2.22 – Tag <p> em funcionamento
Fonte: FIAP (2016)

2.3.2 Tag

Vem da palavra breakline, ou seja, quebra de linha, utilizado antes para “pular” de linha. Esta tag não possui versão de fechamento (</br>).

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de quebra de linha</title>
    <meta charset="utf-8">
  </head>
  <body>
    Linha
    Linha
    Linha3
  </body>
</html>
```

1

2

Figura 2.23 – Tag
 aplicada ao HTML
Fonte: Elaborado pelo autor (2016)



Figura 2.24 – Tag
 em funcionamento
Fonte: FIAP (2016)

2.3.3 Tags <h1>, <h2>, <h3> ... até <h6>

Trata-se de tags específicas para títulos e subtítulos, sendo possível, portanto, subdividir conteúdos até um sexto nível.

Veja o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo h1 até h6</title>
    <meta charset="utf-8">
  </head>
  <body>
    <h1>1. Título</h1>
    <p>Introdução do capítulo</p>

    <h2>1.1. Subtítulo</h2>
    <p>Introdução desta seção</p>

    <h3>1.1.1. Subtítulo</h3>
    <p>Introdução desta subseção</p>
  </body>
</html>
```

Figura 2.25 – Uso das tags <h1>, <h2> e <h3> aplicada ao HTML
Fonte: Elaborado pelo autor (2016)



Figura 2.26 – Tags <h1>, <h2> e <h3> em funcionamento
Fonte: FIAP (2016)

Repare que o tamanho e estilos das tags <h1> até <h6> são diferentes, <h1> possuindo um tamanho maior que o <h2>, que tem um tamanho maior que <h3> e assim por diante.

Muitos diagramadores HTML acabam usando, por exemplo, <h3> como título, sem que haja um <h2> e <h1> em seu documento, julgando o tamanho dos anteriores grande demais para seu documento.– Trata-se de um equívoco: existe uma hierarquia entre estas tags, ou seja, se o documento possui um <h4>, significa que temos antes dele <h1> até <h3>. Se, por acaso, a estilização padrão de <h1> for grande demais para você, altere-a usando CSS (veremos como fazer isso adiante).

2.3.4 Tag

É utilizada para aplicar imagens ao documento HTML. Para isso, conta com seus principais atributos:

- **src:** utilizado para informar o caminho da imagem. Pode ser um caminho relativo, como explicado antes, ou um caminho absoluto, apontando até mesmo para imagens que estejam em outro domínio de internet. Esta abordagem não é muito recomendada, por várias razões: você possui autorização de uso desta imagem? Além disso, mesmo que haja uma autorização, como ter certeza que a imagem estará sempre disponível (e não seja apagada) já que não se tem o controle deste domínio? É sempre recomendado, portanto, que as imagens estejam hospedadas no mesmo local que os documentos HTML em geral.
- **width e height:** trata-se do comprimento e altura da imagem, respectivamente. Muitos diagramadores utilizam estes atributos para realizar um redimensionamento da imagem, aumentando-a (com resultados precários, deixam a imagem “pixelada”) ou diminuindo-a, o que é ainda mais grave: usar uma imagem grande demais e redimensiona-la no HTML não torna a imagem de origem menor; o arquivo, pesado, terá que ser baixado pelo usuário, causando problemas de performance. Aconselha-se a criar arquivos diferente da mesma imagem em dimensões diferentes, quantas forem necessárias para uso.

Para que serve **width** e **height** então? Diagramação. Se, por acaso, a imagem de origem não for localizada, estes atributos garantem que o comprimento e altura sejam reservados para o local que a imagem ocuparia, evitando que haja uma quebra de diagramação do documento. Os atributos devem, portanto, ser informados sempre com o comprimento e altura reais da imagem.

- **alt:** texto alternativo da imagem, importantíssimo para a acessibilidade de um documento. É este texto que é apresentado se

a imagem de origem não for localizada; o texto assume o lugar da imagem. É este mesmo texto que será utilizado pelo leitor de tela de um deficiente visual, que poderá receber uma descrição da imagem que não pode ver.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de imagem</title>
    <meta charset="utf-8">
  </head>
  <body>
      </body>
  </html>
```

Figura 2.27 – Tag aplicada ao HTML
Fonte: Elaborado pelo autor (2016)



Figura 2.28 – Tag em dois momentos: Sem localizar a imagem origem e com a mesma localizada

Fonte: FIAP (2016)

2.3.5 Tag <div>

A tag <div> representa uma divisão ou seção do documento HTML. Trata-se de um container utilizado para agrupar elementos como texto e imagens. Com o CSS, este container pode ser posicionado em qualquer parte da página (esteticamente dizendo), tornando-se uma verdadeira revolução na diagramação de páginas.

Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de div</title>
    <meta charset="utf-8">
  </head>
  <body>
    <div class="artigo">Conteúdo Teste representando um
artigo.</div>
  </body>
</html>
```

Figura 2.29 - Tag <div> aplicada ao HTML
Fonte: Elaborado pelo autor (2016)

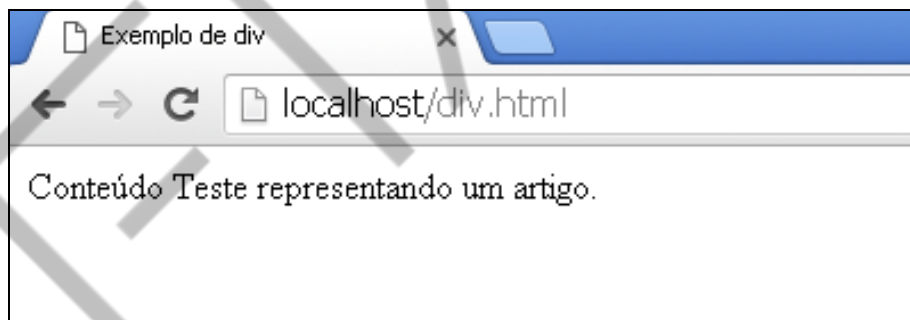


Figura 2.30 – Tag <div> em funcionamento
Fonte: FIAP (2016)

Abordaremos as questões de posicionamento e estilização do texto com detalhes no capítulo de CSS.

2.3.6 Substitutivos do <div> para melhorar a semântica

Muitos autores encorajam fortemente o uso das novas *tags* que foram criadas no HTML5 (como <article>, <section>, entre outras), reservando a *tag*

`<div>` apenas nos casos em que estas novas tags não parecem ser aplicáveis. Todas elas se comportam como a tag `<div>`, mas facilitam a manutenção dos documentos, além de melhorar a compreensão do texto por parte dos mecanismos de busca. São elas:

- **`<section>`**: representa a seção de um documento ou página, podem ser capítulos ou seções de uma tese.
- **`<nav>`**: um container de links para navegação, como um menu.
- **`<article>`**: contém o artigo propriamente dito, trata-se do conteúdo próprio do documento, podendo ser um artigo de jornal ou revista, uma postagem de fórum, entre outros.
- **`<aside>`**: este container ficará ao lado, tangenciando o conteúdo principal (geralmente um artigo), pode ser usado para a publicidade ou um conteúdo de menor importância, pois o `<aside>` é um dos primeiros elementos a sumir em dimensões de tela menores, quando nos referimos a um layout responsivo.
- **`<hgroup>`**: é o título da seção, por ser um agrupamento, pode conter tags `<h1>` até `<h6>`, contendo assim subtítulos, por exemplo.
- **`<header>`**: container de cabeçalho do documento (ou de um artigo), pode possuir menu de navegação (`<nav>`), o título (com elemento `<hgroup>` e/ou `<h1>`-`<h6>`), entre outros.
- **`<footer>`**: geralmente utilizados no final de uma seção, representam o rodapé de uma seção ou do documento como um todo, podem conter links relacionados, referências de direitos autorais, entre outros.
- **`<time>`**: representa não só a hora, mas uma data do calendário gregoriano; é muito útil aos mecanismos de busca, pois indica a idade da informação ali contida (considerado que será usado em um contexto, como um artigo).
- **`<mark>`**: serve para dar ênfase ou destaque em uma parte do texto ou documento, como referência.

- **<figure>**: indica a presença de uma figura no contêiner, podendo conter uma legenda.
- **<figcaption>**: legenda de uma figura, deve ser usado dentro da tag <figure>.

```

<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de Título</title>
    <meta charset="utf-8">
  </head>
  <body>
    <article>Conteúdo Teste representando um
artigo.</article>
  </body>
<!DOCTYPE html>
<html lang="pt-br">
  <header>
    <meta charset="utf-8">
  </header>
  <body>
    <article>
      <header>
        <h1>Título do Artigo</h1>
        <p>Publicado em: <time
pubdate="pubdate" datetime="2015-11-17">17 de novembro de
2015</time></p>
      </header>
      <p>Aqui começa este grande artigo...</p>
      <section>
        <h2>Subseção do Artigo</h2>
        <p>Aqui entra uma subseção...</p>
      </section>
      <section>
        <h2>Outra subseção do Artigo</h2>
        <p>Aqui entra outra subseção...</p>
        <figure>
          
          <figcaption>Figura importante da
seção</figcaption>
        </figure>
      </section>
      <footer>
        <p>Creative Commons Attribution-
ShareAlike License</p>
      </footer>
    </article>
  </body>
</html>

```

Figura 2.31 – Exemplo de uso das novas tags semânticas
Fonte: Elaborado pelo autor (2016)

2.3.7 Tag <a>

Utilizado para fazer uma âncora de hyperlink, ou seja, o hyperlink propriamente dito, tão tradicional na Internet. A seguir, seus principais atributos:

- **href:** *hyperlink reference*, ou seja, o endereço do documento em que o usuário seja submetido caso clique no link. Pode ser um endereço relativo (apontando para o mesmo servidor de hospedagem) ou absoluto, apontando para links externos (outros domínios). Nos primórdios, a *tag* era utilizada para fazer um link dentro do mesmo documento, para um ponto mais adiante (ou mesmo para trás), usando âncoras. Esta abordagem caiu em desuso, sendo preferível por várias razões quebrar um documento HTML muito grande em vários documentos menores (embora seja mais trabalhoso).
- **download:** a presença deste atributo na *tag* <a> indica que o autor deseja que o usuário baixe o documento (seja ele HTML ou não) ao invés de ser submetido a ele.
- **rel:** utilizado para qualificar o relacionamento entre os documentos vinculados. Vide material específico de HTML5 para detalhes.
- **target:** define o alvo do hyperlink. O valor padrão é “_self”, ou seja, ao clicar no hyperlink, a página ou documento informado em href será carregado na mesma janela em que o documento atual está; outras possibilidades são “_blank”, cujo destino é aberto em uma nova janela ou aba, “_parent”, documento abre no frame pai (quando há uso de frames), “_top”, documento de destino ignora a divisão de frames da tela e abre em toda a área de janela disponível, e, finalmente, podemos informar o nome do frame, o qual desejamos que a página ou documento destino seja aberto.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de hyperlink</title>
    <meta charset="utf-8">
  </head>
  <body>
    <a href="outra pagina">Link para outra página</a>
  </body>
</html>
```

Figura 2.32 – Tag <a> implementada em HTML
Fonte: Elaborado pelo autor (2016)

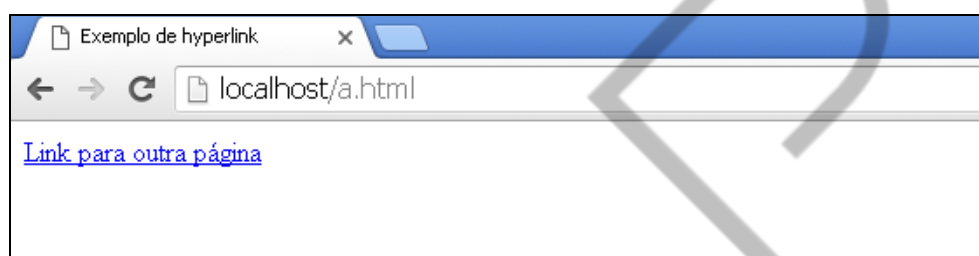


Figura 2.33 – Tag <a> em funcionamento
Fonte: FIAP (2016)

Imagens podem ser utilizadas como hyperlink, combinando as tags <a> e :

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de hyperlink</title>
    <meta charset="utf-8">
  </head>
  <body>
    <a href="http://www.fiap.com.br/">
      
    </a>
  </body>
</html>
```

Figura 2.34 – Tags <a> e criando um hyperlink em uma imagem
Fonte: Elaborado pelo autor (2016)

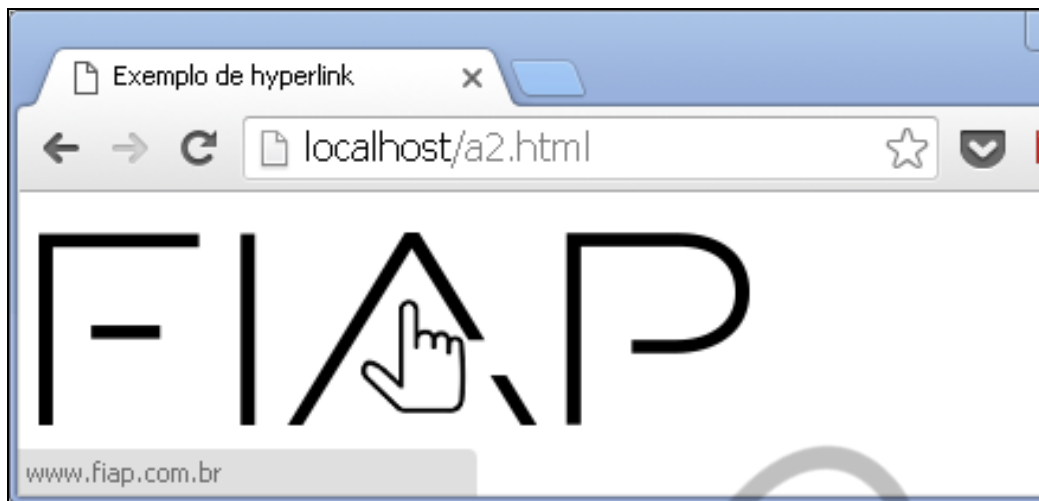


Figura 2.35 – Hyperlink aplicado em uma imagem
Fonte: FIAP (2016)

2.3.8 Tags , e

As tags e são utilizadas para a criação de listas ordenadas e não ordenadas, respectivamente. A tag é utilizada para os itens da lista, independente do formato.

Exemplo de lista ordenada:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de ol</title>
    <meta charset="utf-8">
  </head>
  <body>
    <ol>
      <li value="1">Teste 1</li>
      <li value="2">Teste 2</li>
      <li value="3">Teste 3</li>
    </ol>
  </body>
</html>
```

Figura 2.36 – Tags e criando uma lista ordenada
Fonte: Elaborado pelo autor (2016)

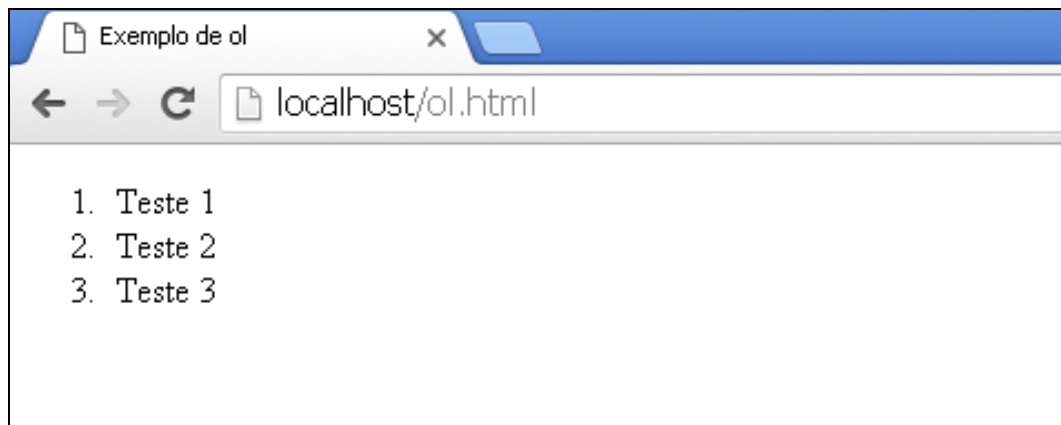


Figura 2.37 – Lista ordenada
Fonte: FIAP (2016)

Exemplo de lista não ordenada:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de ul</title>
    <meta charset="utf-8">
  </head>
  <body>
    <ul>
      <li>Teste 1</li>
      <li>Teste 2</li>
      <li>Teste 3</li>
    </ul>
  </body>
</html>
```

Figura 2.38 – Tags e criando uma lista não ordenada
Fonte: Elaborado pelo autor (2016)

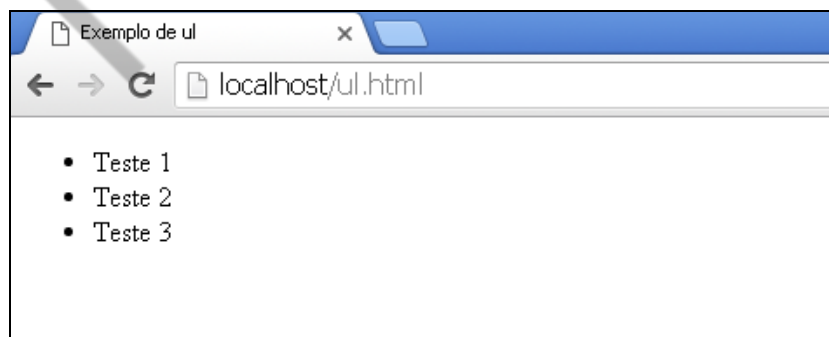


Figura 2.39 – Lista não ordenada
Fonte: FIAP (2016)

2.3.9 Tag <iframe>

Permite criar uma janela dentro do documento HTML na qual é possível abrir outro documento HTML. Os principais atributos são o comprimento da janela (width), sua altura (height), o nome da janela (name) e o endereço internet deste outro documento (src).

Veja o exemplo a seguir, ele precisa de 3 documentos HTML para funcionar: iframe_pagina1.html, iframe_pagina2.html e iframe.html. Este último possui um iframe que carregará previamente a primeira página, e possui um link com target: ao clicar, a página 2 será aberta dentro do iframe:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de iframe</title>
    <meta charset="utf-8">
  </head>
  <body>Página 1</body>
</html>
```

Figura 2.40 – O arquivo iframe_pagina1.html
Fonte: Elaborado pelo autor (2016)

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de iframe</title>
    <meta charset="utf-8">
  </head>
  <body>Página 2</body>
</html>
```

Figura 2.41 – O arquivo iframe_pagina2.html
Fonte: Elaborado pelo autor (2016)

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de iframe</title>
    <meta charset="utf-8">
  </head>
  <body>
    <a href="iframe_pagina2.html" target="janela">Clique
aqui para abrir a página 2 dentro do iframe</a>
    <br><br>
    Iframe abaixo:
    <iframe name="janela" width="100" height="100"
src="iframe_pagina1.html">
    </iframe>
  </body>
</html>
```

Figura 2.42 – O arquivo iframe.html
Fonte: Elaborado pelo autor (2016)

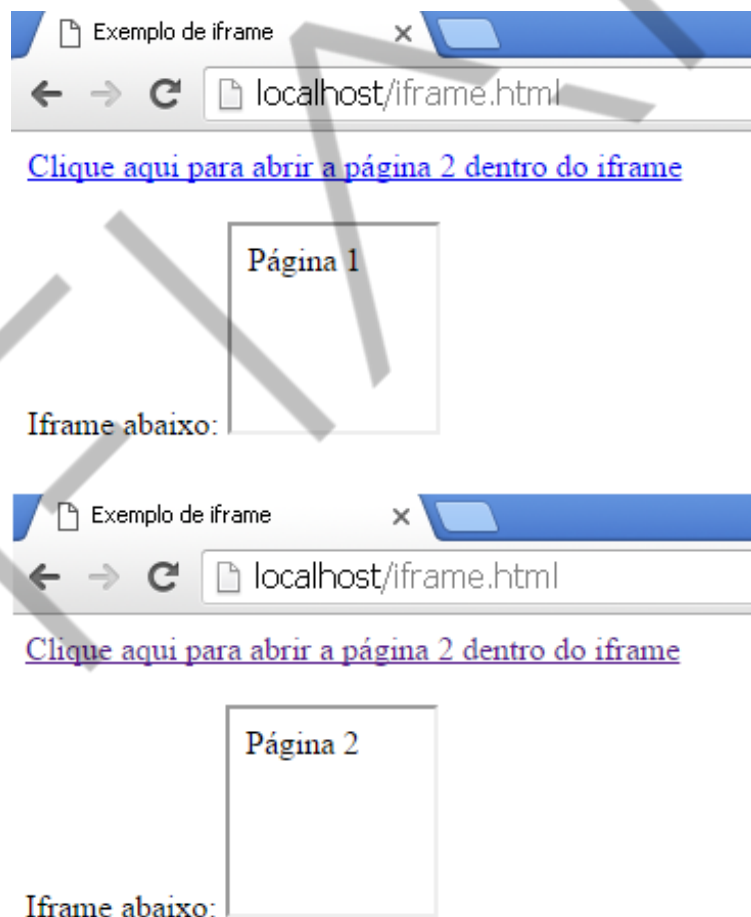


Figura 2.43 – iframe em dois momentos: Na carga da página e ao clicar no hyperlink disponível
Fonte: FIAP (2016)

2.3.10 Tags <frame> e <frameset>: Por que não usar?

As tags em questão permitem dividir uma janela de navegador em vários pedaços, possibilitando a abertura de vários documentos HTML distintos e ao mesmo tempo. Usados à exaustão nas décadas de 1990 e início da década seguinte, pesam demais na performance e são de difícil controle: cada janela é batizada com um nome e os hyperlinks com a tag <a> deve usar o atributo target para abrir conteúdos nas outras janelas.

Em uma Internet acessada em múltiplos dispositivos e com uma preocupação em ser responsiva (abrir o conteúdo adequadamente em diferentes tamanhos de tela) usar frames é um retrocesso.

Se o desenvolvedor quiser reaproveitar todo o layout de um documento HTML, mudando apenas parte dele de tempos em tempos (e sem a necessidade renderizar páginas o tempo todo), é aconselhável que seja feito utilizando JavaScript e AJAX, e isso será abordado no conteúdo devido.

2.3.11 Tags de marcação a se esquecer

Em seus primórdios, existiam *tags* importantes para a marcação de texto, como das tags (formatação de fontes de texto), <big> (fonte maior), <small> (fonte menor), (negrito), <i> (itálico), <s> (tachado) e <u> (sublinhado). Com exceção de , <i> e <u>, todas as *tags* mencionadas foram descontinuadas no HTML5. A razão é que a melhor forma de se estilizar texto se chama CSS: mais organizado; e a possibilidade de reuso dos estilos. Esqueça que estas *tags* existem (ou existiram): utilize a *tag* <div> ou uma *tag* conhecida como com o atributo class e deixe tudo a cargo do CSS, como você aprenderá a fazer no capítulo devido.

2.4 Tabulação de dados: criando tabelas

Desde os primórdios das teses em hipertexto, uma necessidade se mantém constante: a de tabular dados para apresentação. Esta é a única boa

razão para o uso de tabelas. Digo isso porque antes da popularização da *tag* `<div>` e evolução do CSS, tabelas eram utilizadas para diagramar o layout da página, por permitir personalizar suas dimensões e colocar tabelas umas dentro de outras. A esta reformulação (deixar de usar tabelas e passar a usar *div's* posicionados com CSS) damos o nome de *tableless*.

Não quer dizer, no entanto, que tabelas são más e devem ser enterradas a sete palmos. Elas devem ser usadas para o que se propõem: tabular dados!

2.4.1 Tag `<table>`

Utilizar para determinar o início e o fim (`</table>`) de uma tabela. Embora possamos utilizar os atributos **width** e **height** para dimensionar a tabela – seja em seu tamanho exato em pixels ou percentualmente – além de alinhá-la usando **align** e a possibilidade de determinar a espessura de sua borda com **border**, tudo isso pode ser feito com melhores resultados usando CSS. Os exemplos a seguir utilizarão os atributos, mas procure fazê-las usando os estilos que o CSS lhe oferece.

Eis o primeiro exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="100%" height="200">
    </table>
  </body>
</html>
```

Figura 2.44 – Tag `<table>` sendo utilizada
Fonte: Elaborado pelo autor (2016)

E temos o seguinte resultado:

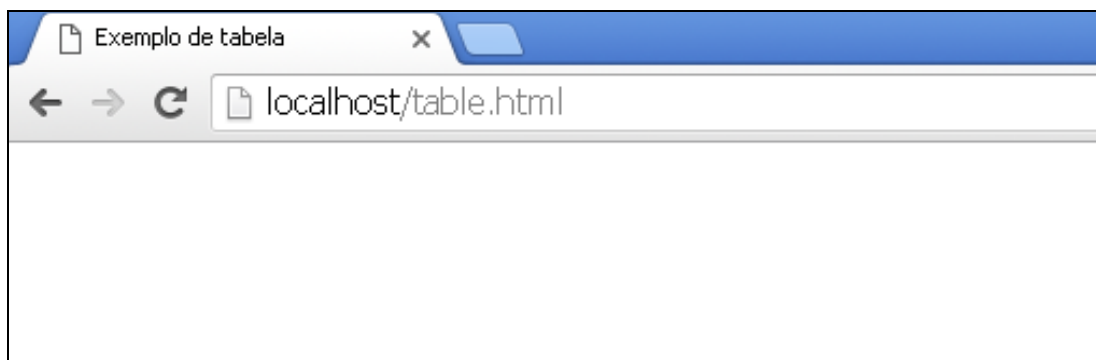


Figura 2.45 – Tabela criada
Fonte: FIAP (2016)

Exatamente! Nada! Sei que é frustrante, mas não temos muito que ver por enquanto... precisamos de outras *tags* para termos algum resultado.

2.4.2 Tags <thead>, <tbody> e <tfoot>

Embora sejam *tags* consideradas opcionais, ajudam a definir bem as áreas distintas de uma tabela. <thead> indica o início do cabeçalho da tabela, <tbody> seu corpo e <tfoot> seu rodapé. Usá-los possibilita, por exemplo, determinar que as duas primeiras linhas da tabela são seu cabeçalho, e não apenas a primeira.

Veja o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="100%" height="200">
      <thead>
      </thead>
      <tbody>
      </tbody>
      <tfoot>
      </tfoot>
    </table>
  </body>
</html>
```

Figura 2.46 – Tags <thead>, <tbody> e <tfoot> sendo utilizadas
Fonte: Elaborado pelo autor (2016)

Segundo a especificação, as *tags* de finalização são consideradas opcionais. Sendo assim, o navegador compreende que o cabeçalho <thead> termina assim que ele encontra um <tbody>, e o corpo da tabela termina assim que encontra o <tfoot>, que termina com o final da tabela:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="100%" height="200">
      <thead>
      <tbody>
      <tfoot>
    </table>
  </body>
</html>
```

Figura 2.47 – Tags <thead>, <tbody> e <tfoot> sem suas *tags* de finalização
Fonte: Elaborado pelo autor (2016)

Embora codificar assim sempre me dá a impressão que está faltando alguma coisa...

2.4.3 Tag <tr>

Utilizada para determinar o início e o fim (</tr>) de uma linha da tabela, que poderá ter inúmeras linhas:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="100%" height="200">
      <thead>
        <!-- Primeira linha -->
        <tr>
        </tr>
        <!-- Segunda linha -->
        <tr>
        </tr>
      </thead>
      <tbody>
        <!-- Terceira linha -->
        <tr>
        </tr>
        <!-- Quarta linha -->
        <tr>
        </tr>
      </tbody>
      <tfoot>
        <!-- Quinta linha -->
        <tr>
        </tr>
      </tfoot>
    </table>
  </body>
</html>
```

Figura 2.48 – Linhas de tabela sendo definidas com a tag <tr>
Fonte: Elaborado pelo autor (2016)

Neste exemplo temos duas no cabeçalho, duas no corpo da tabela e uma no rodapé, e nada ainda para ver. Tenha paciência!

2.4.4 Tags <td> e <th>

Ambas são utilizadas para determinar o início e o fim de uma célula de tabela. Mas você se pergunta: “e as colunas?”. Bem, uma tabela HTML possui tantas colunas quanto sua linha que possui o maior número de tags <td>.

Sendo assim, se em uma linha de tabela temos 3 <td>'s e na segunda 4 <td>'s, a tabela possui 4 colunas, e teremos uma lacuna na primeira.

Tags <td> e <th> são similares: <td> devem ser usados em células comuns, enquanto <th> deve ser usado para células de cabeçalho. A princípio, não há diferença visual entre elas (embora alguns navegadores deixem a fonte dentro de um <th> em negrito), mas você pode fazê-las com CSS, veremos isso. Entretanto, o uso do <th> traz outros benefícios, como repetir o cabeçalho automaticamente ao se mandar imprimir uma grande tabela.

Eis o primeiro exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo          de          tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="500" border="1">
      <thead>
        <tr><!-- Primeira linha -->
          <th>Curso</th>
          <th>Conteudista</th>
          <th>Tutor</th>
          <th>Realizado</th>
        </tr>
      </thead>
      <tbody>
        <tr><!-- Segunda linha -->
          <td>Gamificação</td>
          <td>Henrique Poyatos</td>
          <td>Henrique Poyatos</td>
          <td>Sim</td>
        </tr>
        <tr><!-- Terceira linha -->
          <td>Prototipação</td>
          <td>Almir Alves</td>
          <td>Almir Alves</td>
          <td>Sim</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

Figura 2.49 – Exemplo de tabela usando <td> e <th>
Fonte: Elaborado pelo autor (2016)

Ufa! Finalmente consigo lhe mostrar algo:



Curso	Conteudista	Tutor	Realizado
Gamificação	Henrique Poyatos	Henrique Poyatos	Sim
Prototipação	Almir Alves	Almir Alves	Sim

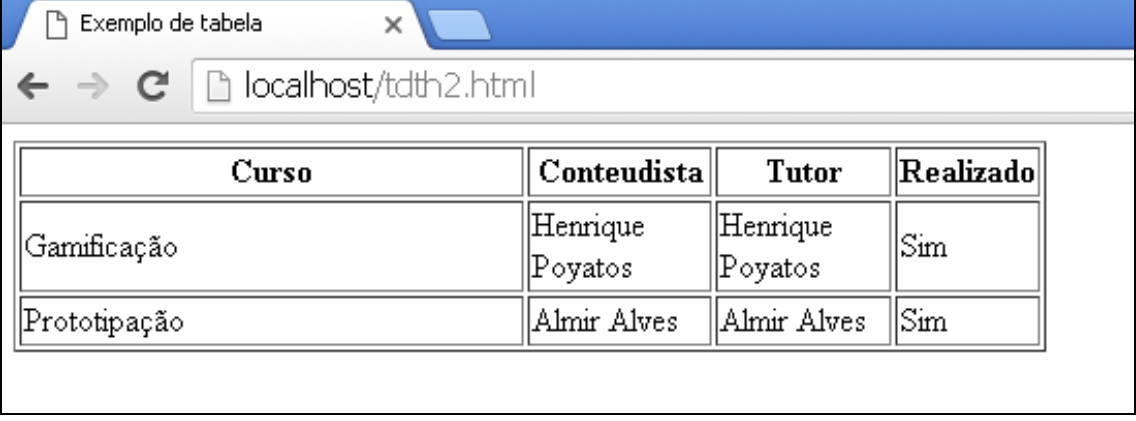
Figura 2.50 – Tabela completa
Fonte: FIAP (2016)

2.4.5 Dimensionando células

É possível mudar a dimensão das células (e, por consequência, das colunas). Se quisermos que a primeira coluna possua 50% do tamanho total da tabela, basta colocar **width="50%"** em um dos <td>'s ou <th>'s que se apresentam logo após os <tr>'s (primeira célula da linha, portanto):

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="500" border="1">
      <thead>
        <tr><!-- Primeira linha -->
          <th width="50%">Curso</th>
          <th>Conteudista</th>
          <th>Tutor</th>
          <th>Realizado</th>
        </tr>
      </thead>
      <tbody>
        <tr><!-- Segunda linha -->
          <td>Gamificação</td>
          <td>Henrique Poyatos</td>
          <td>Henrique Poyatos</td>
          <td>Sim</td>
        </tr>
        <tr><!-- Terceira linha -->
          <td>Prototipação</td>
          <td>Almir Alves</td>
          <td>Almir Alves</td>
          <td>Sim</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

Figura 2.51 – Mudando as dimensões de uma célula (e coluna)
Fonte: Elaborado pelo autor (2016)



The screenshot shows a web browser window with the title 'Exemplo de tabela' and the address bar displaying 'localhost/tdth2.html'. The browser content area contains a table with the following data:

Curso	Conteudista	Tutor	Realizado
Gamificação	Henrique Poyatos	Henrique Poyatos	Sim
Prototipação	Almir Alves	Almir Alves	Sim

Figura 2.52 – Dimensionando a primeira célula (e coluna) em 50% do tamanho total da tabela
Fonte: FIAP (2016)

Existe um certo limite, no entanto. Se colocarmos a primeira célula (e coluna) em 70% ou 80%, as palavras “Conteudista” e “Realizado” não permitiram “espremer” a segunda e quarta colunas ainda mais; o navegador faria a primeira no máximo percentual possível.

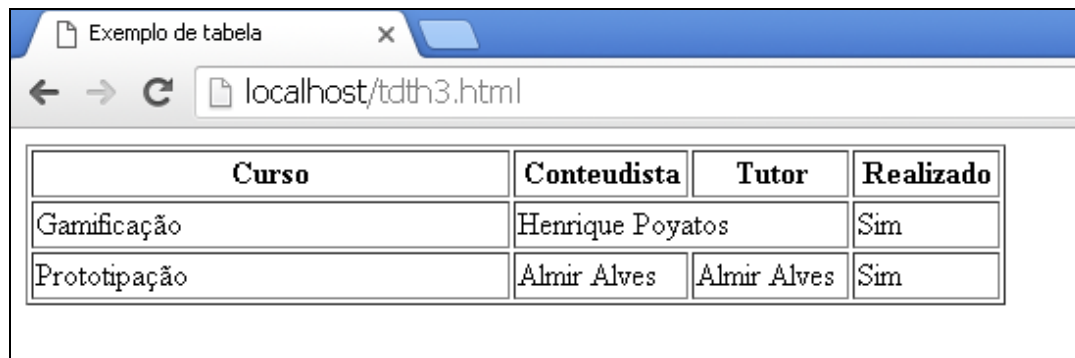
2.4.6 Mesclando colunas

As tags `<td>` e `<th>` possuem atributos que permitem mesclá-las com outras: São o **colspan** (para mesclar colunas, horizontalmente) e **rowspan** (para mesclar linhas, na vertical).

Se quisermos, por exemplo, mesclar as colunas de Conteudista e Tutor nas células que se repetem, como “Henrique Poyatos”? Para isso, precisamos colocar um **colspan="2"** no segundo `<td>` da segunda linha. Além disso, precisamos **eliminar** o terceiro `<td>` da segunda linha. Sim, isso mesmo: o segundo `<td>` agora vale por dois:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="500" border="1">
      <thead>
        <tr><!-- Primeira linha -->
          <th width="50%">Curso</th>
          <th>Conteudista</th>
          <th>Tutor</th>
          <th>Realizado</th>
        </tr>
      </thead>
      <tbody>
        <tr><!-- Segunda linha -->
          <td>Gamificação</td>
          <td colspan="2">Henrique Poyatos</td>
          <td>Sim</td>
          <td><del>Henrique Poyatos</del></td>
        </tr>
        <tr><!-- Terceira linha -->
          <td>Prototipação</td>
          <td>Almir Alves</td>
          <td>Almir Alves</td>
          <td>Sim</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

Figura 2.53 – Mesclando células horizontalmente
Fonte: Elaborado pelo autor (2016)



The screenshot shows a web browser window with the title 'Exemplo de tabela' and the address bar displaying 'localhost/tdth3.html'. The browser displays a table with the following structure:

Curso	Conteudista	Tutor	Realizado
Gamificação	Henrique Poyatos		Sim
Prototipação	Almir Alves	Almir Alves	Sim

Figura 2.54 – Mesclando células horizontalmente

Fonte: FIAP (2016)

Algo similar é feito ao mesclar as duas colunas com a palavra “Sim”.

Desta vez, usaremos rowspan:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de tabela</title>
    <meta charset="utf-8">
  </head>
  <body>
    <table width="500" border="1">
      <thead>
        <tr><!-- Primeira linha -->
          <th width="50%">Curso</th>
          <th>Conteudista</th>
          <th>Tutor</th>
          <th>Realizado</th>
        </tr>
      </thead>
      <tbody>
        <tr><!-- Segunda linha -->
          <td>Gamificação</td>
          <td colspan="2">Henrique Poyatos</td>
          <td rowspan="2">Sim</td>
        </tr>
        <tr><!-- Terceira linha -->
          <td>Prototipação</td>
          <td>Almir Alves</td>
          <td>Almir Alves</td>
          <td>Sim</td>
        </tr>
      </tbody>
    </table>
  </body>
</html>
```

Figura 2.55 – Mesclando células verticalmente

Fonte: Elaborado pelo autor (2016)



The screenshot shows a web browser window with the title 'Exemplo de tabela' and the address bar displaying 'localhost/tdth4.html'. The browser content shows a table with four columns: 'Curso', 'Conteudista', 'Tutor', and 'Realizado'. The 'Realizado' column has a value 'Sim' that spans two rows, demonstrating vertical cell merging.

Curso	Conteudista	Tutor	Realizado
Gamificação	Henrique Poyatos		Sim
Prototipação	Almir Alves	Almir Alves	

Figura 2.56 – Mesclando células verticalmente

Fonte: FIAP (2016)

2.5 Criando formulários

A principal forma de interação, no que diz respeito à entrada de dados, são os formulários. Desde as primeiras versões do HTML, são a forma mais eficaz (quando não era a única) de solicitar informações do usuário.

2.5.1 Tag <form>

Utilizada para determinar o início e o fim de um formulário HTML. Todas as *tags* seguintes estarão contidas no formulário, ou seja, entre <form> e </form>. Além dos atributos globais, os listados a seguir estão disponíveis para parametrização:

- **accept-charset:** utilizada para armazenar o padrão de caracteres que será usado ao submeter o formulário; procure utilizar utf-8 e, em último caso, iso-8859-1.
- **action:** utilizada para armazenar o endereço internet (URL) no qual o formulário será submetido. Embora aceite caminhos absolutos (além dos relativos), o ideal é que esta URL de tratamento do formulário esteja em um mesmo domínio, caso contrário, teremos uma falha de segurança conhecida como **cross-origin**.
- **autocomplete:** permite que as funcionalidades de autocompletar presentes no navegador auxiliem o usuário no preenchimento do formulário.

- **enctype:** define qual tipo de codificação será utilizada para submeter o formulário. O padrão é “application/x-www-form-urlencoded”, opção a partir da qual os dados são preparados para fazer parte da URL (método GET de envio). As outras possibilidades são “multipart/form-data”, necessário quando requerer a realização de upload de arquivos, usando o formulário (os dados se tornam binários) e “text/plain”, na qual espaços se tornam sinais de adição (“+”) mas nenhum outro tratamento é aplicado.
- **method:** define qual método HTTP será utilizado para submeter o formulário. Existem duas opções: GET e POST.

O método GET, embora seja o padrão, não é a situação ideal. Ao utilizá-lo, ele passa as informações do formulário no endereço URL seguinte. Isso implica em limitações no tamanho da informação passada, além de possíveis problemas de segurança.

No método POST, as informações são enviadas do corpo de mensagem do protocolo HTTP, possibilitando volumes maiores de informação, ocultando (um pouco) a informação dos usuários.

Nota: A não ser que o protocolo usado seja HTTPS (estabelece um túnel criptografado entre cliente e servidor), seja GET ou POST, as informações são enviadas de forma aberta, podendo ser interceptadas por qualquer nó de rede, na qual os pacotes HTTP são enviados. Conclusão, métodos POST são menos inseguros do que métodos GET.

- **name:** define o nome que será dado ao formulário dentro da API `document.forms` (JavaScript). Útil, mas dê preferência para o atributo `id`, que é mais prático.
- **novalidate:** se este atributo estiver presente em `<form>` (não precisa de valor definido), a validação do formulário (que acontece antes da submissão) é ignorada.
- **target:** o mesmo atributo apresentado na tag `<a>`.

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form          id="formulario"          method="POST"
action="processaForm.php">
    </form>
  </body>
</html>
```

Figura 2.57 – Exemplo de aplicação da tag <form> em HTML
Fonte: Elaborado pelo autor (2016)

A exemplo do que aconteceu com as tabelas, ainda não há muito o que se ver.

2.5.2 Tag <label>

Trata-se do rótulo do campo, uma descrição do que aquele elemento de formulário solicita ou representa. Ele se torna eficaz quando utilizado com o atributo for: Este permite associar o <label> com seu elemento de formulário (seja <input>, <select>, <textarea>..), bastando informar o id do elemento. Exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form          id="formulario"          method="POST"
action="processaForm.php">
      <label for="nome">Nome Completo</label>
      <input type="text" id="nome"><!-- explicado na
sequência -->
    </form>
  </body>
</html>
```

Figura 2.58 – Exemplo de uso da tag <label> em HTML
Fonte: Elaborado pelo autor (2016)

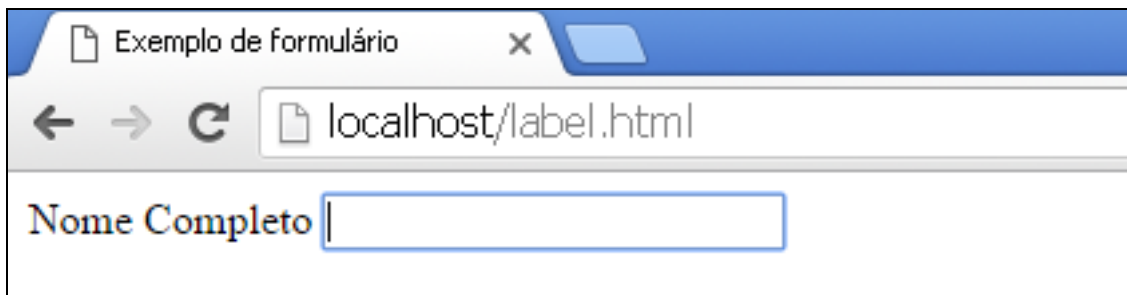


Figura 2.59 – Rótulo para campos de formulário
Fonte: FIAP (2016)

Ao clicar no `<label>` “Nome completo” com o mouse, a caixa de texto ganha foco imediatamente.

2.5.3 Tag `<input>`

É o principal elemento de um formulário, e pode assumir a forma de uma caixa de texto ou um botão de seleção única ou múltipla, um botão de formulário, entre outras possibilidades. Seus principais atributos são:

- **accept:** funciona como dica de quais tipos de arquivo serão aceitos em um elemento de upload (`type="file"`).
- **alt:** o mesmo apresentado em ``, utilizado no caso de `type="image"`.
- **autocomplete:** habilita a funcionalidade de autopreenchimento.
- **autofocus:** concede o foco ao elemento automaticamente, ao carregar a página; apenas um dos elementos de formulário pode possuir este atributo.
- **checked:** ao colocar este atributo em um elemento que seja `type="checkbox"` ou `type="radio"`, o elemento vem marcado como padrão; do contrário, ele vem desmarcado.
- **disabled:** ao colocar este atributo no elemento, indica que ele está indisponível no momento, e fica indisponível para alterações.
- **height:** utilizado para definir a altura do elemento. Prefira seu equivalente em CSS.

- **list:** informa uma lista de opções para o recurso de autocompletar (utiliza a tag `<datafield>`, a ser estudada adiante).
- **max:** novo atributo, define o valor máximo que um campo do tipo numérico ou data (formato yyyy-mm-dd) pode assumir.
- **maxlength:** tamanho máximo, em caracteres, esperado para o valor preenchido.
- **min:** novo atributo, define o valor mínimo que um campo do tipo numérico ou data (formato yyyy-mm-dd) pode assumir.
- **minlength:** tamanho mínimo, em caracteres, esperado para o valor preenchido.
- **multiple:** permite ao campo coletar vários valores de uma vez. Campo múltiplo.
- **name:** define o nome do elemento na API `form.elements` (JavaScript). Este elemento é particularmente importante em uma linguagem como PHP, cujas API de tratamento também usam nome. Defina também `id`, pois é mais prático ao usar JavaScript.
- **pattern:** novo atributo, permite definir um padrão em expressões regulares para validar este campo de formulário.
- **placeholder:** trata-se de um rótulo visível para o usuário, posicionado dentro do elemento. Geralmente é utilizado como dica de preenchimento.
- **readonly:** ao colocar este atributo no elemento, indica que ele é apenas leitura, e fica indisponível para alterações.
- **required:** novo atributo, ao informá-lo indica que o campo é de preenchimento obrigatório.
- **size:** tamanho do campo em número de caracteres. Prefira usar `width` em CSS.
- **type:** define qual é o tipo de elemento de formulário. As opções são:
 - **text:** caixa de texto padrão.

- **hidden:** campo “escondido” ou “invisível”. Não é renderizado.
- **password:** caixa de texto que mascara os caracteres digitados.
- **checkbox:** uma caixa para seleção múltipla.
- **radio:** um botão redondo para seleção única.
- **file:** uma caixa de texto e botão que permite procurar arquivos no repositório local do usuário. Campo para *upload* de arquivos.
- **submit:** um botão que, ao ser clicado, submete o formulário.
- **reset:** um botão que, ao ser clicado, restaura o formulário para os seus valores iniciais, ou seja, geralmente apaga o formulário.
- **button:** um botão que não faz nada ao ser clicado; é específico para ser trabalhado usando JavaScript.

Existem vários tipos novos criados no HTML5:

- **tel:** específico para armazenar telefone.
- **url:** específico para armazenar um endereço web (URL).
- **search:** específico para campo de pesquisa.
- **email:** específico para armazenar um endereço de e-mail, valida automaticamente.
- **number:** específico para armazenar valores numéricos.
- **date:** específico para armazenar data.
- **time:** específico para armazenar hora.
- **range:** específico para intervalo numérico.

- **color:** específico para armazenar cores.
- **value:** permite definir um valor padrão para o campo. Desta maneira, ao renderizar o formulário, o campo vem previamente preenchido.
- **width:** utilizado para definir o comprimento do elemento. Prefira seu equivalente em CSS.

O exemplo abaixo utiliza os atributos value e disabled:

```
<!DOCTYPE                                html>
<html                                    lang="pt-br">
  <head>
    <title>Exemplo                        de                formulário</title>
    <meta                                charset="utf-8">
  </head>
  <body>
    <form                                id="formulario"                method="POST">
      <label                            for="nome">Nome                Completo:                </label>
      <input                            type="text"                id="nome"                value="João                Ninguém"
```

Figura 2.60 – Exemplo de formulário usando <input> com value e disabled
Fonte: Elaborado pelo autor (2016)

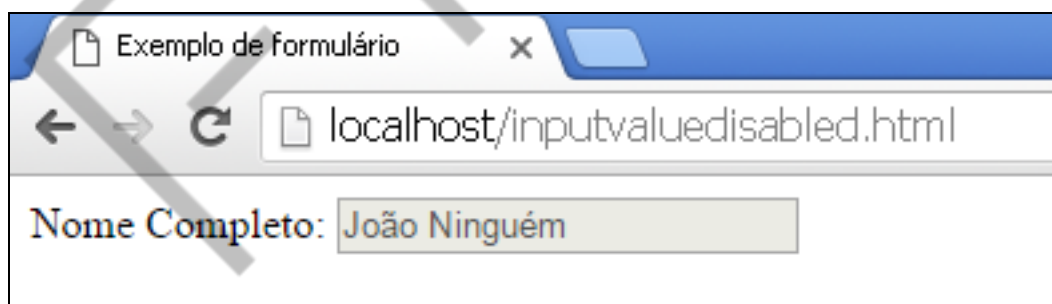


Figura 2.61 – Exemplo de formulário usando <input> com value e disabled
Fonte: FIAP (2016)

No exemplo seguinte, botões de seleção única e caixas de múltipla seleção usando o atributo checked:

```

<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <!-- Botões de escolha única -->
      <label for="estcivil">Estado Civil: </label>
      <input type="radio" name="estcivil"
value="Solteiro">Solteiro
      <input type="radio" name="estcivil"
value="Casado">Casado
      <input type="radio" name="estcivil" value="Viúvo"
checked>Viúvo<br>

      <!-- Caixas de múltipla escolha -->
      <label for="interesses">Interesses: </label>
      <input type="checkbox" name="interesses"

```

Figura 2.62 – Exemplo de formulário usando <input> com checked
Fonte: Elaborado pelo autor (2016)

Exemplo de formulário

localhost/inputchecked.html

Estado Civil: ☐ Solteiro ☐ Casado ☒ Viúvo

Interesses: ☐ Música ☒ Cinema ☒ Teatro

Figura 2.63 – Exemplo de formulário usando <input> com checked
Fonte: FIAP (2016)

Repare que os botões de escolha única (type="radio") possuem os mesmos atributos name; é desta maneira que estes botões são agrupados,

permitindo que apenas um deles seja marcado. No caso de caixas de múltipla escolha (`type="checkbox"`), isso é desejável, embora não seja obrigatório.

Veja um exemplo utilizando todos os tipos de `<input>` mais tradicionais:



```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <!-- Caixa de texto -->
      <label for="nome">Nome Completo: </label>

      <!-- Caixa para senha -->
      <label for="senha">Senha: </label>
      <input type="password" id="senha"><br>

      <!-- Botões de escolha única -->
      <label for="estcivil">Estado Civil: </label>
      <input type="radio" name="estcivil"
value="Solteiro">Solteiro
      <input type="radio" name="estcivil"
value="Casado">Casado
      <input type="radio" name="estcivil"
value="Viúvo">Viúvo<br>

      <!-- Caixas de múltipla escolha -->
      <label for="interesses">Interesses: </label>
      <input type="checkbox" name="interesses"
value="Música">Música
      <input type="checkbox" name="interesses"
value="Cinema">Cinema
      <input type="checkbox" name="interesses"
value="Teatro">Teatro<br>

      <!-- Campo para upload -->
      <label for="foto">Foto: </label>
      <input type="file" id="foto"><br>
```

```
<!-- Botões -->
<input type="submit" value="Enviar">
<input type="reset" value="Limpar dados">
<input type="button" value="Faz nada">
</form>
</body>
</html>
```

Figura 2.64 – Exemplo de formulário usando <input>
Fonte: Elaborado pelo autor (2016)

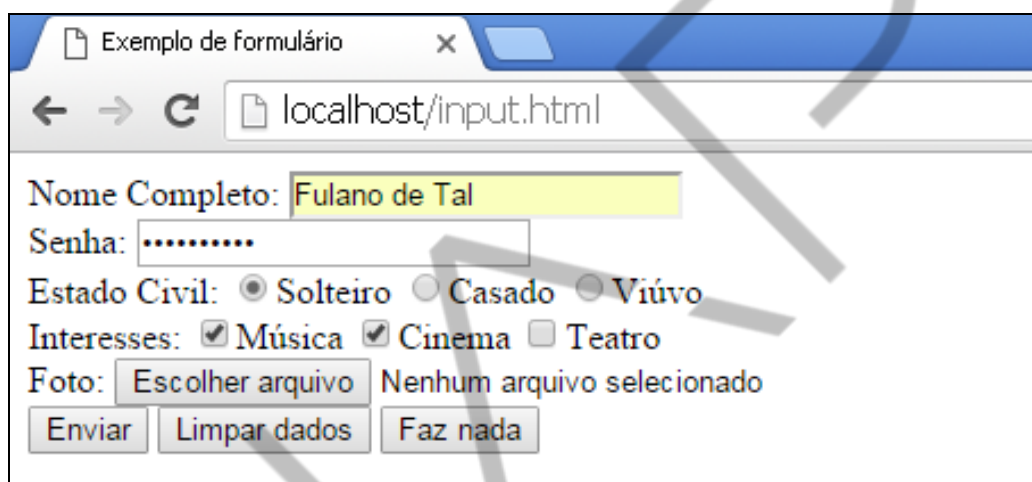


Figura 2.65 – Exemplo de formulário usando <input>
Fonte: FIAP (2016)

Veja o material específico de HTML5 para outros exemplos abordando os novos type's e atributos.

2.5.4 Tags <select>, <option> e <optgroup>

Estas três *tags* são: utilizar para criar caixas de seleção (conhecidas como "ComboBox"). A tag <select> define o início e o fim da lista, que pode ser parametrizada com os seguintes atributos:

- **atributos globais, autofocus, disabled, name e required:** funcionamento idêntico ao explicado na *tag* <input>.
- **multiple:** permite selecionar mais de um valor na caixa de seleção.

- **size:** tamanho na lista em número de linhas, transformando-a em uma lista.

No entanto, <select> precisa conter tags <option> para que faça sentido. Elas correspondem às opções que o usuário terá para escolher nesta caixa de seleção. Os atributos fundamentais são:

- **disabled:** desabilita a opção, tornando-a indisponível.
- **value:** define o valor da opção, ou seja, que valor será submetido caso aquela opção for selecionada.
- **selected:** permite definir uma ou mais opções (caso <select> seja multiple) que começarão previamente selecionados.

Veja um exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <select id="estcivil" name="estcivil">
        <option value="S">Solteiro</option>
        <option value="C">Casado</option>
        <option value="V" selected>Viúvo</option>
        <option value="D">Divorciado</option>
      </select>
    </form>
  </body>
</html>
```

Figura 2.66 – Exemplo de formulário usando <select> e <option> com opção previamente selecionada

Fonte: Elaborado pelo autor (2016)

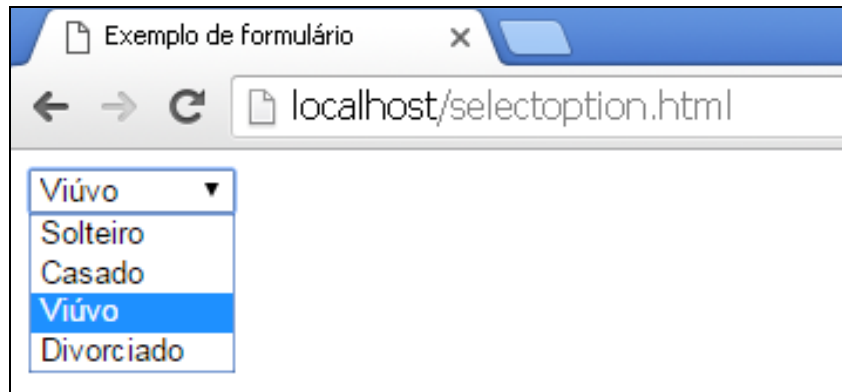


Figura 2.67 – Exemplo de formulário usando `<select>` e `<option>` com opção previamente selecionada

Fonte: FIAP (2016)

Caso este formulário fosse submetido, apenas as iniciais dos estados civis – “S”, “C”, “V”, e “D” seria enviado adiante, bem diferente do rótulo visível ao usuário, contido dentro da tag `<option>`.

Conforme já mencionado, podemos criar listas para múltipla seleção: basta utilizar os atributos `multiple` e `size`. Aproveitemos o exemplo para mostrar o funcionamento da tag `<optgroup>`, que agrupa as opções em categorias:


```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <label for="carros">Carros preferidos</label>
      <select id="carros" name="carros" size="8" multiple>
        <optgroup label="FIAT">
          <option value="500">500</option>
          <option value="Uno">Palio</option>
          <option value="Uno">Uno</option>
        </optgroup>
        <optgroup label="FORD">
          <option value="Fiesta">Fiesta</option>
          <option value="Focus">Focus</option>
          <option value="Ka">Ka</option>
        </optgroup>
      </select>
    </form>
  </body>
</html>
```

Figura 2.68 – Exemplo de formulário usando <select>, <option> e <optgroup> com seleção múltipla

Fonte: Elaborado pelo autor (2016)



Figura 2.69 – Exemplo de formulário usando `<select>`, `<option>` e `<optgroup>` com seleção múltipla

Fonte: FIAP (2016)

2.5.5 Tag `<textarea>`

Utilizado quando é necessário que o usuário escreva um longo texto, grande demais para caixas de texto comuns. Na sequência, os principais atributos:

- **atributos globais, autocomplete, autofocus, disabled, maxlength, minlength, name, placeholder, readonly e required:** funcionamento idêntico ao explicado na tag `<input>`.
- **cols:** define o número de colunas do `<textarea>`, seu tamanho na horizontal (prefira `width` no CSS).
- **rows:** define o número de linhas do `<textarea>`, seu tamanho na vertical (prefira `height` no CSS).

Repare que `<textarea>` não possui o atributo `value`: O valor padrão do campo deve ser contido dentro da tag (ou seja, entre `<textarea>` e `</textarea>`).

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <label for="obs">Observações</label><br>
      <textarea id="obs" rows="5" cols="50">Texto de
exemplo.</textarea>
    </form>
  </body>
</html>
```

Figura 2.70 – Exemplo de formulário usando <textarea>
Fonte: Elaborado pelo autor (2017)

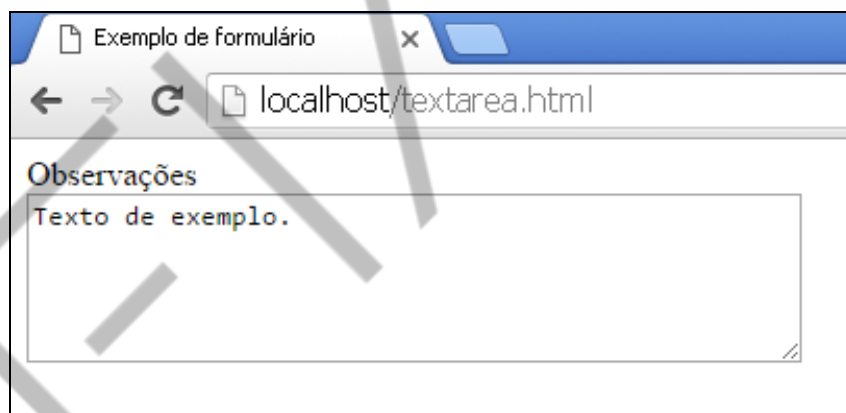


Figura 2.71 – Exemplo de formulário usando <textarea>
Fonte: FIAP (2016)

2.5.6 Tags <fieldset> e <legend>

Estas possibilitam um agrupamento de campos de um formulário.

Veja o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <fieldset>
        <legend>Dados pessoais</legend>
        <label for="nome">Nome Completo:</label>
        <input type="text" id="nome"><br>
        <label for="idade">Idade:</label>
        <input type="number" id="idade"><br>
      </fieldset>
    </form>
  </body>
</html>
```

Figura 2.72 – Exemplo de formulário usando <fieldset> e <legend>
Fonte: Elaborado pelo autor (2016)

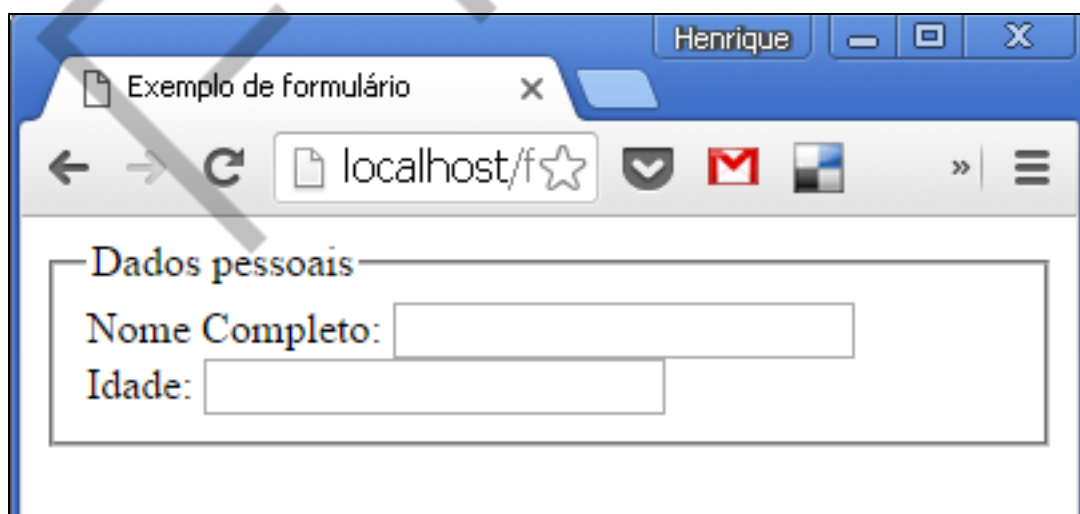


Figura 2.73 – Exemplo de formulário usando <fieldset> e <legend>
Fonte: FIAP (2016)

2.5.7 Tag <datalist>

Novo elemento de formulário, combina <input type="text"> com <select>, ou seja, é uma caixa de seleção que permite digitação. Assim como <select> utiliza <option> para determinar as opções disponíveis. Repare que o <input type="text"> é necessário previamente utilizando o atributo list.

Atenção: A tag <datalist> não possui suporte em navegadores Safari e Internet Explorer versão 9 e anteriores.

Eis o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <label for="navegador">Qual seu navegador preferido?</label>
      <input list="navegadores" name="navegador">
      <datalist id="navegadores">
        <option value="Google Chrome">Google Chrome</option>
        <option value="Mozilla Firefox">Mozilla Firefox</option>
        <option value="Internet Explorer">Internet Explorer</option>
        <option value="Opera">Opera</option>
        <option value="Apple Safari">Apple Safari</option>
      </datalist>
    </form>
  </body>
</html>
```

Figura 2.74 – Exemplo de formulário usando <datalist>
 Fonte: Elaborado pelo autor (2016)

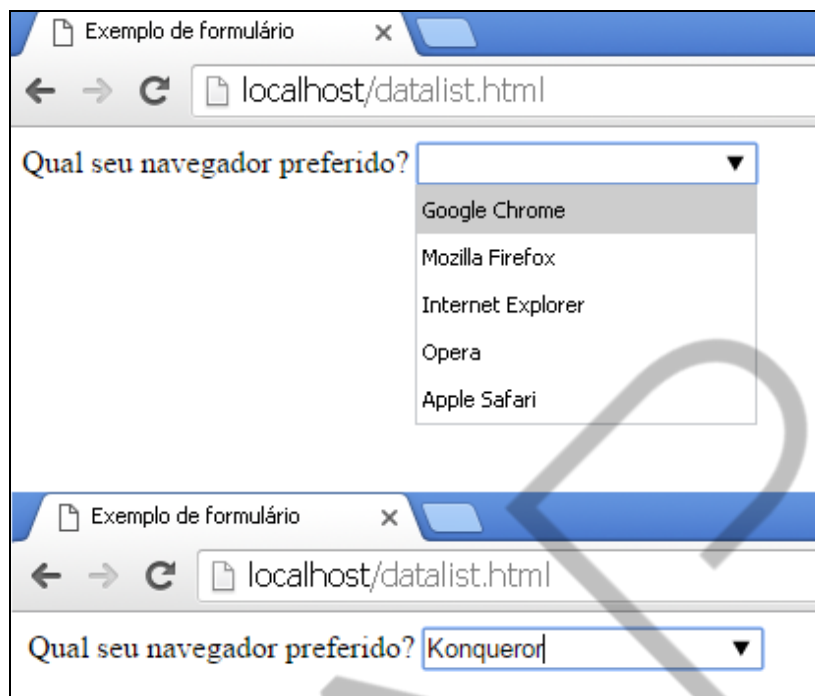


Figura 2.75 – Exemplo de formulário usando <datalist> em dois momentos: usando como caixa de seleção e digitando uma opção não encontrada na lista
 Fonte: FIAP (2016)

2.5.8 Tag <progress>

Novo recurso que possibilita mostrar o progresso de alguma ação. Geralmente utilizado para mostrar o progresso de um upload, utiliza os atributos value (valor atual) e max (valor total). Veja o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Exemplo de formulário</title>
    <meta charset="utf-8">
  </head>
  <body>
    <form id="formulario" method="POST">
      <progress value="25" max="100"></progress>
    </form>
  </body>
</html>
```

Figura 2.76 – Exemplo de formulário usando <progress>
Fonte: Elaborado pelo autor (2016)

Atenção: a tag <progress> não possui suporte em navegadores Internet Explorer versão 9 e anteriores.

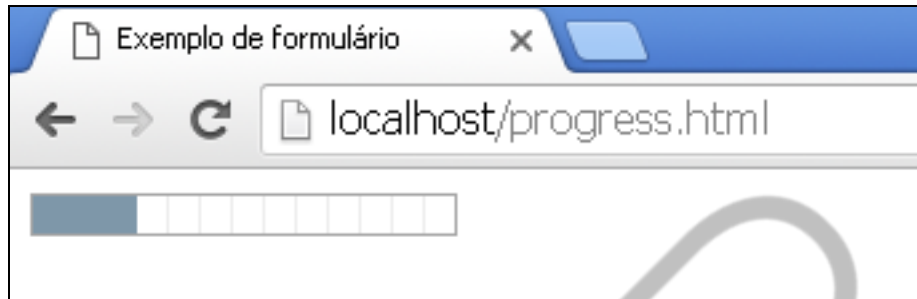


Figura 2.77 – Exemplo de formulário usando <progress>
Fonte: FIAP (2016)

2.6 Considerações finais

O objetivo deste conteúdo foi abordar o básico e essencial da linguagem HTML; sendo assim, existem diversas *tags* e atributos que foram desconsiderados, seja por estarem descontinuados ou por terem pouco uso prático. Existem ainda outros usos para a *tag* <input> e *tags* importantes como <audio> e <video> que são abordados em nosso conteúdo específico de HTML5, recomendamos a leitura.

REFERÊNCIAS

W3C. **HTML 4.01 Specification.** [s.d.]. Disponível em: <<https://www.w3.org/TR/html4/>>. Acesso em: 22 jan. 2016.

_____. **HTML 5 Specification.** [s.d.]. Disponível em: <<https://www.w3.org/TR/html5/>>. Acesso em: 22 jan. 2016.

_____. **XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition).** [s.d.]. Disponível em: <<https://www.w3.org/TR/xhtml1/>>. Acesso em: 22 jan. 2016.

_____. **Extensible Markup Language (XML) 1.1 (Second Edition).** [s.d.]. Acesso em: 22 jan. 2016.

W3Counter. **December 2015 Market Share.** [s.d.]. Disponível em: <<http://www.w3counter.com/globalstats.php>>. Acesso em: 22 jan. 2016.

EMERSON