

# Unidade II

## 3 FORMULÁRIOS COM HTML

Vamos continuar nossos estudos de elaboração de páginas web, ainda utilizando como base os recursos do HTML5. Veremos, neste tópico, os conceitos de criação de formulários. Os formulários permitem que automatizemos o envio e o recebimento de dados de uma página. Eles são úteis para logins de sites e fóruns, assinatura de livros de visitas, e-mail baseado na web, questionários on-line etc. Apresentaremos, em um primeiro momento, algumas tags e alguns atributos importantes na criação de formulários para, em seguida, mostrarmos exemplos de aplicação.

### 3.1 Elemento principal de um formulário: `<form> ... </form>`

Todos os elementos de um formulário, como campos de entrada de dados e botões, devem estar dentro das tags `<form>` e `</form>`. Na maioria dos casos, um formulário deve ter os atributos **name**, **action** e **method** definidos conforme exposto a seguir.

- **name=""**: recebe como valor um nome exclusivo que identifica o formulário, que pode ser utilizado pelo script de ação.
- **action="url"**: define o endereço (URL – uniform resource locator ou localizador de padrão de recursos) do script que processará os dados do formulário quando enviado. Em alguns casos, o URL da ação não é necessário; por exemplo, quando uma função JavaScript do cliente é programada na página da web para processar os dados do formulário.
- **method=""**: o atributo method define o método HTTP de submissão dos dados do servidor. Os valores utilizados são get ou post.
  - Método post: não tem limitação quanto à quantidade de dados e não mostra os dados na linha de endereço do browser, sendo mais robusto e seguro do que o get. Poderia ser utilizado, por exemplo, para enviar dados para um formulário de registro do usuário.
  - Método get: é utilizado para enviar uma pequena quantidade de dados que não necessitam de sigilo de envio, pois, nesse caso, os dados enviados ficam visíveis na linha de endereço. Pode ser aplicado, por exemplo, em pesquisas que devem retornar informações não sigilosas.

### 3.1.1 Utilizando o método get

O método get é o método HTTP padrão, que pode ser utilizado ao enviar dados de formulário. Conforme já mencionado, quando o get é utilizado como valor do atributo method, todos os dados do formulário ficam visíveis no campo de endereço da página web.

Exemplo: `/action_page.htm?disciplina=DSI&unidade=2`

Veja, a seguir, algumas recomendações importantes.

- Anexar todos os dados do formulário ao URL nos pares nome/valor que transitam na URL.
- Verificar que o tamanho de um URL é limitado (2.048 caracteres).
- Evitar a utilização do get para enviar dados confidenciais, pois ficará visível na URL.
- Observar que se trata de algo útil para envios de formulários nos quais um usuário possa efetuar uma marcação de resultado como favorito.
- Notar que o get pode ser considerado eficaz para dados não seguros, como cadeias de consulta no Google, que auxiliam na busca e obtenção de dados.

### 3.1.2 Utilizando o método post

Vale reforçar: sempre devemos utilizar o método post quando os dados do formulário puderem conter informações confidenciais ou pessoais. Assim, ele não apresentará os dados do formulário no campo de endereço da página web.

Veja, a seguir, algumas recomendações importantes.

- Observar que o post não tem limitações de tamanho, e, por isso, podemos ter a sua utilização para enviar grandes quantidades de dados na web.
- Verificar que os envios de formulários com post não possibilitam ser marcados como favoritos.

Exemplo: `<form action="unip.br" method="post">`

No quadro a seguir, ilustramos alguns atributos do elemento `<form>` que podem ser utilizados para facilitar a construção de um formulário HTML.

**Quadro 8 – Atributos para serem utilizados no form**

Atributo	Descrição
<i>accept-charset</i>	Especifica o conjunto de caracteres usados no formulário enviado (padrão: o conjunto de páginas)
<i>action</i>	Especifica um endereço (URL) para o qual enviar o formulário (padrão: a página de envio)
<i>autocomplete</i>	Especifica se o navegador deve preencher automaticamente o formulário (padrão: ativado)
<i>enctype</i>	Especifica a codificação dos dados enviados (padrão: é codificado por URL)
<i>method</i>	Especifica o método HTTP usado ao enviar o formulário (padrão: GET)
<i>name</i>	Especifica um nome usado para identificar o formulário. Este atributo é utilizado para referenciar elementos em JavaScript ou referenciar dados de um formulário após sua submissão
<i>novalidate</i>	Especifica que o navegador não deve validar o formulário
<i>target</i>	Especifica o destino do endereço no atributo action (padrão: <code>_self</code> )

## 3.2 Campo de entrada: <input>

O elemento <input> é utilizado para criar um campo de entrada de dados do usuário no formulário. Ele pode gerar tanto um campo de entrada de texto simples quanto entradas mais complexas. Veja uma lista de alguns importantes atributos do elemento <input>.

- **name=""**: recebe como valor um nome exclusivo para a entrada, que pode ser utilizado pelo script de ação.
- **type=""**: seu valor define o tipo de entrada de dados. Existem vários tipos de campos de entrada de formulário. A seguir, listamos alguns tipos muito comuns.
  - Text: quando o elemento <input> possui o atributo **type** configurado com o valor text, o campo de entrada de dados recebe um texto simples do usuário.
  - Password: permite que o usuário digite um valor que não é exibido de forma legível, sendo utilizado para a entrada de senhas. Nesse caso, o elemento <form> deve ter o atributo method definido com o valor post, para que a senha não seja mostrada na linha de endereço.
  - Checkbox: define uma caixa de seleção, que pode ser marcada ou desmarcada pelo usuário. Nesse caso, o usuário é capaz de selecionar quantas caixas quiser entre as disponíveis. Por exemplo, ao perguntarmos ao usuário quais linguagens de programação ele domina, podemos apresentar opções selecionáveis por meio de caixas de seleção, esperando que ele selecione todas as que se enquadrem em seu perfil.
  - Radio: define um *radio button*, que apresenta uma opção ao lado de um círculo selecionável. Botões desse tipo diferenciam-se de um *checkbox* porque, no caso, só é possível selecionar uma entre as opções disponíveis de seu grupo. Por exemplo, podemos perguntar o nível de escolaridade do usuário dessa forma, apresentando as opções por meio de *radio buttons*. Ele será capaz de selecionar apenas uma opção.

- **File**: permite ao usuário a seleção de um arquivo para upload.
- **Image**: define uma imagem como botão de submissão de dados. O usuário deve clicar nessa imagem para realizar o envio de seus dados.
- **Hidden**: com esse valor, o campo não aparece na tela para o usuário. Pode ser utilizado para transportar dados do cliente para o servidor para a identificação do formulário.
- **value=""**: seu valor corresponde ao dado inicialmente exibido no campo de entrada quando o formulário é carregado pela primeira vez.
- **size=""**: define o tamanho ou a largura da entrada, normalmente designada em termos de caracteres numéricos, em vez de pixels.
- **maxlength=""**: comprimento máximo do campo de entrada, como o número máximo de caracteres para uma entrada de texto.
- **checked**: usado com uma entrada do tipo *checkbox*. Ele define que a caixa apareça para o usuário previamente selecionada.



### Saiba mais

Você consegue testar todos os valores disponíveis para o atributo `type` do elemento `<input>` no site indicado a seguir.

W3SCHOOLS. *HTML <input> type Attribute*. 2022. Disponível em: <https://bit.ly/38SpUTt>. Acesso em: 5 maio 2022.

### 3.3 Botão: `<button>`

O `<button>` é um elemento que cria um botão de formulário, o qual deve realizar alguma ação ao ser acionado. De seu conjunto de atributos, listamos os que seguem.

- **name=""**: nome exclusivo para o botão a ser usado pelo script de ação.
- **type=""**: o tipo de botão, `submit` ou `reset`, determina se o formulário deve ser enviado (`submit`) ou limpo (`reset`) após pressioná-lo.
- **value=""**: texto que aparece no botão, como "OK" ou "Enviar".
- **size=""**: determina o comprimento (ou largura) do botão.

### 3.4 Lista suspensa de seleção: `<select> ... </select>`

O elemento `<select>` cria uma lista suspensa (do inglês, *drop-down list*), também chamada de caixa de combinação, que, geralmente, permite a seleção de um item entre os vários itens disponíveis na lista. Dos seus atributos, destacamos os que seguem.

- **name=""**: nome do seletor.
- **size=""**: tamanho mínimo (largura) da lista de seleção. Geralmente não é necessário, pois o tamanho dos itens definirá o tamanho da lista.
- **multiple**: permite que um usuário selecione vários itens da lista. Normalmente, só é possível selecionar uma opção.

A criação dos itens que integram a lista suspensa será comentada no tópico a seguir.

#### 3.4.1 Item de seleção da lista suspensa: `<option> </option>`

Um elemento `<option>` é necessário para cada item da lista suspensa, e deve aparecer dentro das marcas `<select>` e `</select>`. O texto a ser mostrado para a opção deve aparecer entre as marcas `<option>` e `</option>`. Veja alguns atributos dessa tag.

- **value=""**: recebe o valor, ou seja, os dados enviados para o script de ação com a opção selecionada. Esse não é o texto que aparece na lista.
- **selected**: define a opção padrão que é selecionada automaticamente quando o formulário é mostrado.

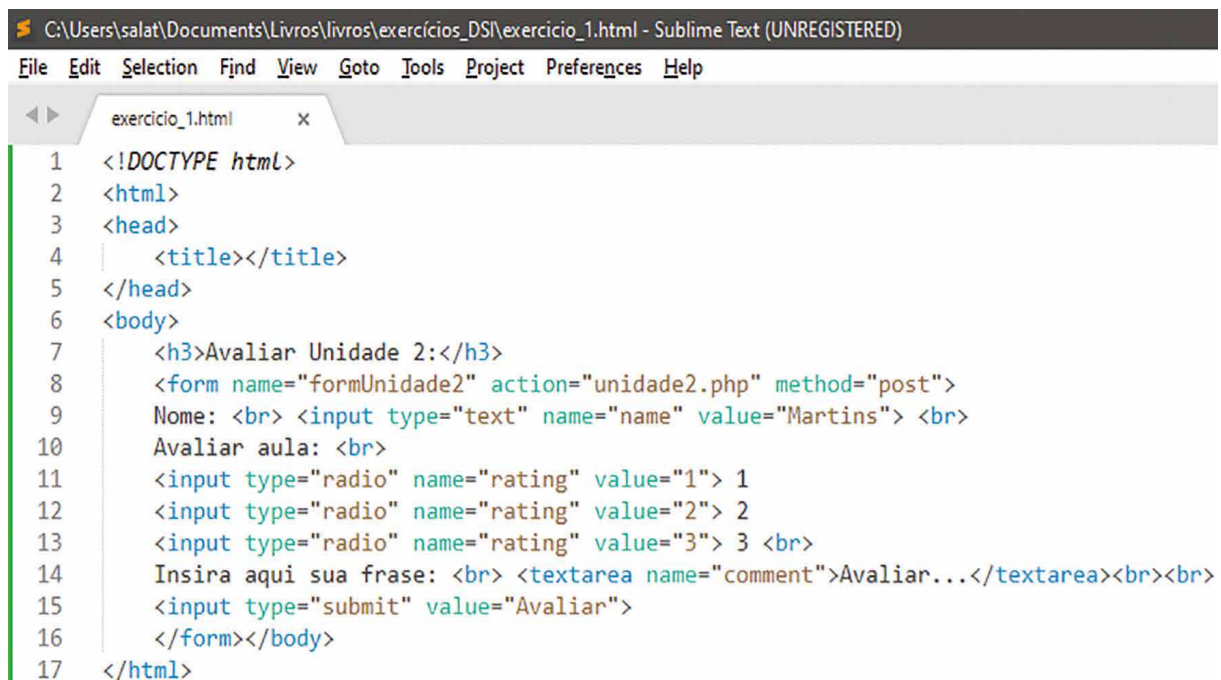
### 3.5 Área de texto: `<textarea> </textarea>`

O elemento `<textarea>` cria uma entrada que permite que uma grande quantidade de texto seja inserida, possibilitando que a altura da caixa de entrada seja especificada, diferentemente do elemento `<input>` padrão. Veja alguns atributos de interesse desse elemento.

- **name=""**: nome exclusivo atribuído ao campo do formulário.
- **rows=""**: número de linhas de texto, que define o tamanho vertical da área de texto.
- **cols=""**: tamanho horizontal da caixa de texto, definido como o número de caracteres (ou seja, colunas).

#### Exemplo de aplicação

Vamos agora construir um formulário simples em HTML, para colocar em prática os nossos conhecimentos. Considere uma aplicação de formulários, conforme ilustrado na figura a seguir.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <title></title>
5 </head>
6 <body>
7   <h3>Avaliar Unidade 2:</h3>
8   <form name="formUnidade2" action="unidade2.php" method="post">
9     Nome: <br> <input type="text" name="name" value="Martins"> <br>
10    Avaliar aula: <br>
11    <input type="radio" name="rating" value="1"> 1
12    <input type="radio" name="rating" value="2"> 2
13    <input type="radio" name="rating" value="3"> 3 <br>
14    Insira aqui sua frase: <br> <textarea name="comment">Avaliar...</textarea><br><br>
15    <input type="submit" value="Avaliar">
16  </form></body>
17 </html>
```

Figura 69 – Exemplo de construção de formulário em HTML

Para criar o formulário ilustrado, é necessário fazer o que se explica a seguir.

- Abrir o Sublime Text.
- Criar um documento HTML.
- Dentro do elemento <body>, digitar as tags:

```
<h3>Avaliar a Unidade 2: </h3>
```

Isso cria o cabeçalho "Avaliar a Unidade 2:".

```
<form name="formUnidade2 action=unidade2.php method="post">
```

Isso cria um formulário formUnidade2, que, por meio do método post, enviará as informações preenchidas quando o botão submit for acionado para o arquivo unidade2.php. Não abordaremos aqui como o servidor processa a submissão de um formulário. Esse tema costuma ser abordado em conteúdos que tratam da linguagem PHP, muito utilizada para comunicação do lado do servidor (*back-end*).

```
<input type="text" name="name" value="Martins">
```

Cria-se uma caixa de texto name e, dentro dela, fixa-se o texto "Martins", que é o valor inicial adotado para a caixa. O texto, obviamente, pode ser modificado pelo usuário antes do envio do formulário.

```
<input type="radio" name="rating" value="1">
```

Criamos as opções de 1 a 3 para selecionar por meio do *radio button*. Esse passo deve ser feito três vezes, mudando o campo `value=""`, conforme vemos na figura. Os inputs que possuem o mesmo nome farão parte do mesmo grupo. Dessa forma, as opções 1, 2 e 3, que possuem nome "rating", farão parte do grupo de avaliação da aula, e apenas uma delas poderá ser selecionada pelo usuário.

```
<textarea name="comment">Avaliar...</textarea>
```

Cria-se uma caixa de texto para informar o comentário. O texto padrão dessa caixa será "Avaliar...", que pode ser modificado pelo usuário antes do envio.

```
<input type="Submit" value="Avaliar">
```

Cria-se um botão para submeter os dados preenchidos para outra página ou um banco de dados.

A execução do nosso arquivo HTML no navegador deverá ter um aspecto parecido com o apresentado na figura a seguir. Assim, temos o resultado da construção de um formulário e seus componentes.

### Avaliar Unidade 2:

Nome:

Avaliar aula:

☒ 1 ☐ 2 ☐ 3

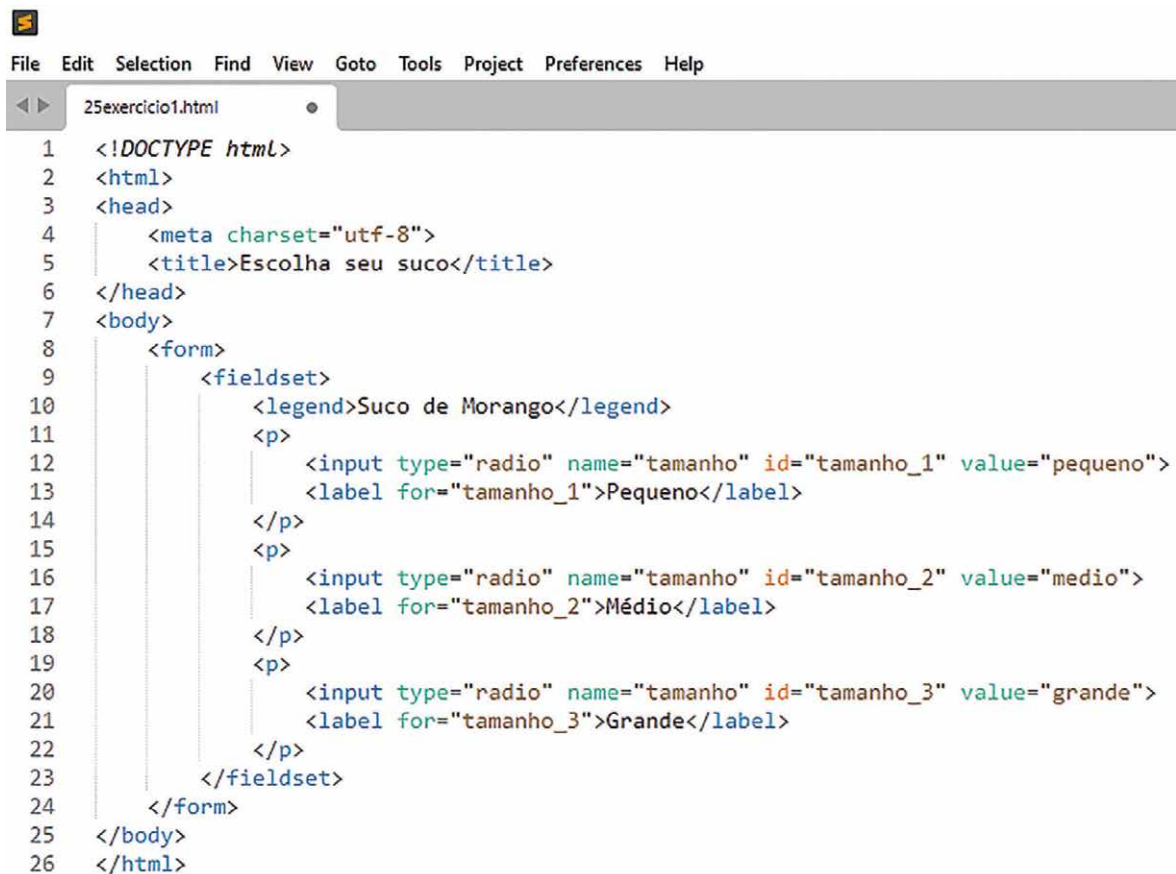
Insira aqui sua frase:

Figura 70 – Página exibida no browser: formulário

### 3.6 Elementos <fieldset> e <legend>

O elemento <fieldset> é utilizado para agrupar elementos relacionados em um formulário, desenhando um quadro em volta deles. O elemento <legend> pode ser usado para definir um rótulo para os elementos do <fieldset>.

Podemos ver exemplos de aplicação de formulários com *legend* e *fieldset* na figura a seguir.



```

1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Escolha seu suco</title>
6  </head>
7  <body>
8      <form>
9          <fieldset>
10             <legend>Suco de Morango</legend>
11             <p>
12                 <input type="radio" name="tamanho" id="tamanho_1" value="pequeno">
13                 <label for="tamanho_1">Pequeno</label>
14             </p>
15             <p>
16                 <input type="radio" name="tamanho" id="tamanho_2" value="medio">
17                 <label for="tamanho_2">Médio</label>
18             </p>
19             <p>
20                 <input type="radio" name="tamanho" id="tamanho_3" value="grande">
21                 <label for="tamanho_3">Grande</label>
22             </p>
23         </fieldset>
24     </form>
25 </body>
26 </html>
    
```

Figura 71 – Exemplo de aplicação de *fieldset* e *legend*

Na figura anterior, foi implementado um elemento `<fieldset>`, responsável pelo desenho de uma caixa envolvendo todos os componentes adicionados a ela, tais como `<input>` e `<label>`. Adicionou-se um elemento `<legend>` para registrar o rótulo da caixa, que, no caso, é "Suco de Morango".

O elemento `<label>` define um rótulo para elementos, como o input de formulário. Para a utilização do `<label>`, criamos um id para cada um dos nossos inputs. Esses ids são posicionados em cada rótulo criado, de forma a identificá-los por meio do atributo **for**. Com a criação de rótulos, o usuário pode clicar no próprio texto para selecionar a opção desejada (não é necessário clicar na circunferência de seleção no caso do *radio button*). O resultado da execução do código é mostrado na figura a seguir.



Figura 72 – Página exibida no browser: formulário *legend* e *fieldset*



Na figura a seguir, temos mais um exemplo de formulário HTML. Implementou-se o elemento `<form>` para ser um formulário que envia dados por meio do método post. Todos os componentes adicionados à página, tais como `<input>` e `<textarea>`, serão enviados para outra página ou para um banco de dados por meio da ação do botão Enviar.

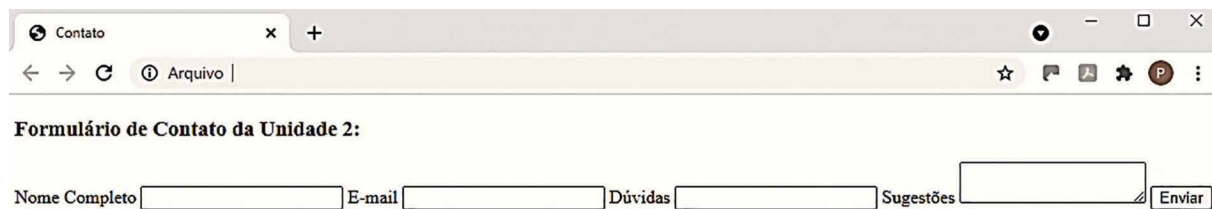
Também utilizamos o elemento `<span>`, responsável por agrupar partes de um texto ou documento em uma linha. O `<span>` funciona de forma semelhante ao `<div>`, que vimos na unidade I. No entanto, o `<div>` é geralmente utilizado para demarcar blocos maiores de código. Note também que o elemento `<label>` pode simplesmente envolver os elementos que ela deve rotular, em vez de utilizar um id com atributo for.



```
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Contato</title>
6  </head>
7  <body>
8      <form name="meuForm" method="post">
9          <h3>Formulário de Contato da Unidade 2:</h3>
10         <label>
11             <span>Nome Completo</span>
12             <input type="text" name="nome">
13         </label>
14         <label>
15             <span>E-mail</span>
16             <input type="text" name="email">
17         </label>
18         <label>
19             <span>Dúvidas</span>
20             <input type="text" name="duvidas">
21         </label>
22         <label>
23             <span>Sugestões</span>
24             <textarea name="sugestoes"></textarea>
25             <input type="button" value="Enviar">
26         </label>
27     </form>
28 </body>
29 </html>
```

Figura 73 – Exemplo de aplicação de formulário com input

O resultado da execução do código é ilustrado na figura a seguir.



Contato x +

Arquivo |

Formulário de Contato da Unidade 2:

Nome Completo  E-mail  Dúvidas  Sugestões

Figura 74 – Página exibida no browser: formulário de contato

## 3.7 Validações de campos de um formulário em HTML

A validação de campos de um formulário corresponde à verificação de alguns pré-requisitos estabelecidos pelo programador. Em certas ocasiões, precisamos tornar impossível o envio do formulário com um campo vazio, por exemplo. Aqui, pretendemos apresentar apenas alguns exemplos simples para você começar a entender os elementos básicos da validação de campos de um formulário. Esses exemplos funcionam nos seguintes browsers: Safari 5, Chrome 6, Opera 9, Firefox 4 Beta e iPhone/iPad. Além disso, é importante ter em mente que cada browser tem um comportamento padrão ligeiramente diferente.



### Saiba mais

Para uma introdução mais detalhada da validação de formulário HTML5, obtenha mais informações na indicação a seguir.

W3SCHOOLS. *HTML Forms*. 2022. Disponível em: <https://bit.ly/3NbDJvI>. Acesso em: 5 maio 2022.

### 3.7.1 Atributo obrigatório

Em termos de validação, a ação mais simples que você pode fazer em seus formulários é marcar um campo de entrada de texto como "obrigatório". Isso pode ser feito utilizando o atributo **required**. Esse é um atributo booleano que, quando presente, especifica que o elemento deve ser preenchido antes do envio do formulário. Observe o exemplo a seguir.

Informe seu nome: `<input type="text" name="name" required>`

Nesse caso, o atributo indica ao navegador que o usuário deve, obrigatoriamente, digitar algo na caixa de texto.



### Lembrete

Um dos métodos mais comuns do HTTP é o get. Quando o utilizamos, os parâmetros são passados no cabeçalho da requisição. Por isso, podem ser vistos pela URI, como no caso do formulário de login `<form method="get">`.

No caso de formulários web, é muito comum que esse método seja o post, que, ao contrário do get, envia os parâmetros no corpo da requisição HTTP, escondendo-os da URI, como no exemplo `<form action="/logar" method="post">`.

O atributo `required` pode ser utilizado nos elementos `<input>`, `<select>` e `<textarea>`. A seguir, vemos um exemplo de código em que esse atributo foi aplicado dentro dos elementos citados.

```
<form action="/action_page.html">
  Nome: <input type="text" name="nome" required>
  <input type="submit">
</form>
<select required>
  <option value="">None</option>
  <option value="azul">Azul</option>
  <option value="amarelo">Amarelo</option>
  <option value="branco">Branco</option>
  <option value="preto">Preto</option>
</select>
<form action="/action_page.html">
  <textarea name="comentario" required></textarea>
  <input type="submit">
</form>
```

Isso informa ao navegador da web (compatível com o HTML5) que o campo deve ser considerado obrigatório. Navegadores diferentes podem ter ações diferentes, como:

- marcar a caixa de entrada de alguma forma (o Firefox 4 Beta adiciona uma sombra de caixa vermelha por padrão);
- exibir um aviso (Opera);
- impedir que o formulário seja enviado se esse campo não tiver valor.

Antes de digitar qualquer coisa na caixa, é exibido um marcador vermelho. Assim que um único caractere é inserido, isso muda para um marcador verde, para indicar que a entrada é válida.

É aqui que o HTML5, por exemplo, realmente se torna interessante e muito útil. Juntamente ao tipo de entrada de texto, agora existem várias outras opções, incluindo e-mail, URL, número, telefone, data e muitas outras.

Alterando o tipo de entrada para e-mail, enquanto também utiliza o atributo necessário, o navegador pode ser usado para validar (de forma limitada) os endereços de e-mail.

Exemplo:

Endereço de e-mail: `<input type="email" name="email" required placeholder="Informar Email válido">`

Os tipos de entrada de número e faixa também aceitam parâmetros para min, max e step. Na maioria dos casos, você pode excluir a etapa, pois o padrão é 1.

Veja o exemplo a seguir.

- **Idade:** `<input type="number" size="6" name="idade" min="18" max="99" value="21"><br>`
- **Satisfação:** `<input type="range" size="2" name="satisfacao" min="1" max="5" value="3">`



### Lembrete

O atributo **required** é booleano e, quando presente, especifica que o elemento deve ser preenchido antes de enviar o formulário.

### 3.7.2 Atributos **readonly** e **disabled**

O atributo **disabled** é um atributo booleano que especifica que um elemento do código HTML deve ser desabilitado. Nesse caso, o elemento torna-se inutilizável. Ele pode ser aplicado para impedir a utilização de um elemento pelo usuário até que algum outro pré-requisito seja cumprido. Os elementos `<input>`, `<button>`, `<select>`, `<fieldset>` e `<textarea>` trabalham com esse atributo.

O atributo **readonly**, por sua vez, também é um operador booleano que especifica que elementos `<input>` ou `<textarea>` sejam do tipo somente leitura. Isso significa que esses elementos não poderão ser modificados, por mais que o usuário consiga realizar outras ações, como selecionar e copiar o texto do campo.

Veja, nas figuras a seguir, exemplos de aplicação de validação HTML que utilizam dentro das tags as propriedades **readonly** e **disabled**. Teste os exemplos em seu navegador e note a diferença na utilização de cada um desses atributos.



File Edit Selection Find View Goto Tools Project Preferences Help

```
28exercicio1.html x
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Unidade 2</title>
6  </head>
7  <body>
8      <form>
9          <label for="nome1">Nome:</label><br>
10         <input type="text" id="nome1" name="nome" value="Martins" readonly><br>
11         <label for="sobrenome1">Sobrenome:</label><br>
12         <input type="text" id="sobrenome1" name="sobrenome" value="Oliveira">
13     </form>
14 </body>
15 </html>
```

Figura 75 – Validação com o campo readonly



File Edit Selection Find View Goto Tools Project Preferences Help

```
29exercicio1.html x
1  <!DOCTYPE html>
2  <html>
3  <head>
4      <meta charset="utf-8">
5      <title>Unidade 2</title>
6  </head>
7  <body>
8      <form>
9          <label for="nome1">Nome:</label><br>
10         <input type="text" id="nome1" name="nome" value="Martins" disabled><br>
11         <label for="sobrenome1">Sobrenome:</label><br>
12         <input type="text" id="sobrenome1" name="sobrenome" value="Oliveira">
13     </form>
14 </body>
15 </html>
```

Figura 76 – Validação com o campo disabled



### Saiba mais

Os validadores HTML são ferramentas capazes de testar se o código que escrevemos atende aos padrões da W3C e segue boas práticas. Podem também buscar por tags ou atributos inválidos, entre outros problemas. O HTML válido é o único contrato que há com os fabricantes de browsers. A especificação diz como devemos escrever documentos e como eles devem interpretá-los. Nos últimos tempos, a conformidade com os padrões dos navegadores atingiu o ponto em que, desde que possamos escrever um código válido, todos os principais browsers devem interpretar nosso código da mesma maneira.

Existe o validador do W3C, que auxilia na identificação e validação de páginas HTML e se encontra na indicação a seguir.

W3C. *Markup Validation Service*. 2013. Disponível em: <http://validator.w3.org/>. Acesso em: 5 maio 2022.

## 4 LISTAS COM HTML E INTRODUÇÃO AO XHTML

Neste tópico, abordaremos mais um conceito das construções de páginas web com HTML, que é a criação de listas. Em sequência, falaremos um pouco de XHTML, que é uma linguagem de marcação baseada em HTML, mas que incorpora recursos XML.

### 4.1 Listas HTML

As listas HTML podem ser utilizadas para apresentar um conjunto de informações de maneira bem formada e semântica. Existem três tipos diferentes de lista em HTML, sendo que cada um tem uma finalidade e significado específicos.

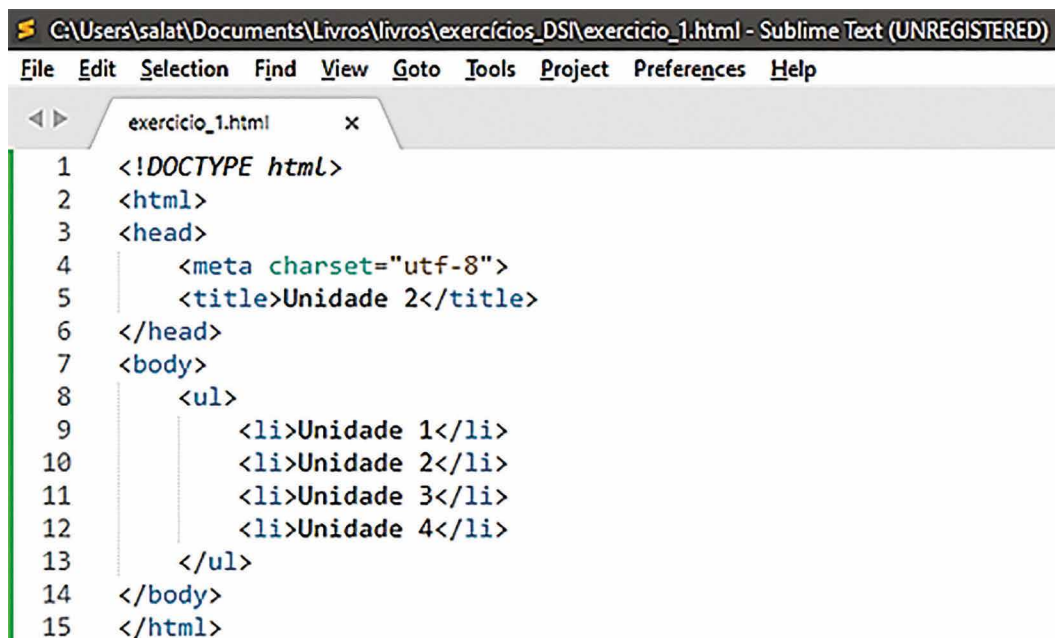
- **Lista não ordenada:** usada para criar uma lista de itens relacionados, em nenhuma ordem específica.
- **Lista ordenada:** usada para criar uma lista de itens relacionados, em uma ordem específica.
- **Lista de descrição:** usada para criar uma lista de termos e suas descrições.

Abordaremos os três tipos nos tópicos seguintes.

#### 4.1.1 Lista não ordenada

Uma lista não ordenada pode ser criada utilizando o elemento `<ul>`. Cada item da lista deve ser envolvido pelo elemento `<li>`. Os itens da lista não ordenada são identificados com marcadores (*bullets*).

As listas não ordenadas são geralmente utilizadas quando a ordem dos itens não é importante. Vamos acompanhar um exemplo, mostrado na figura a seguir.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Unidade 2</title>
6 </head>
7 <body>
8   <ul>
9     <li>Unidade 1</li>
10    <li>Unidade 2</li>
11    <li>Unidade 3</li>
12    <li>Unidade 4</li>
13  </ul>
14 </body>
15 </html>
```

Figura 77 – Exemplo de aplicação de lista não ordenada

A execução do código no navegador resulta no que é ilustrado a seguir. Note a exibição de marcadores no início de cada um dos itens da nossa lista.

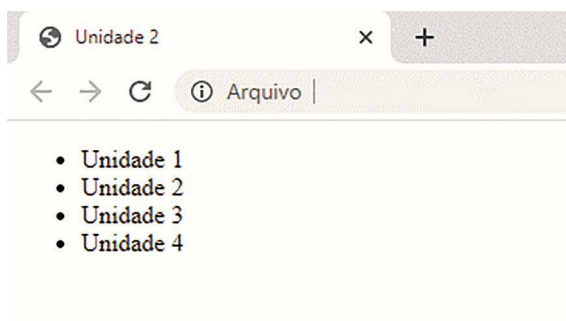


Figura 78 – Página exibida no browser: lista não ordenada

Por padrão, um navegador mostra um *bullet* redondo (tipo *disc*) para uma lista não ordenada. Isso pode ser alterado por meio do atributo **type** do elemento `<ul>`, que mudará o tipo de marcador de todos os itens da lista. Os possíveis valores do atributo `type` são os que seguem.

- **Disc**: marcador padrão, representando o bullet redondo.
- **Circle**: marcador representado por uma circunferência não preenchida.
- **Square**: marcador representado por um pequeno quadrado.



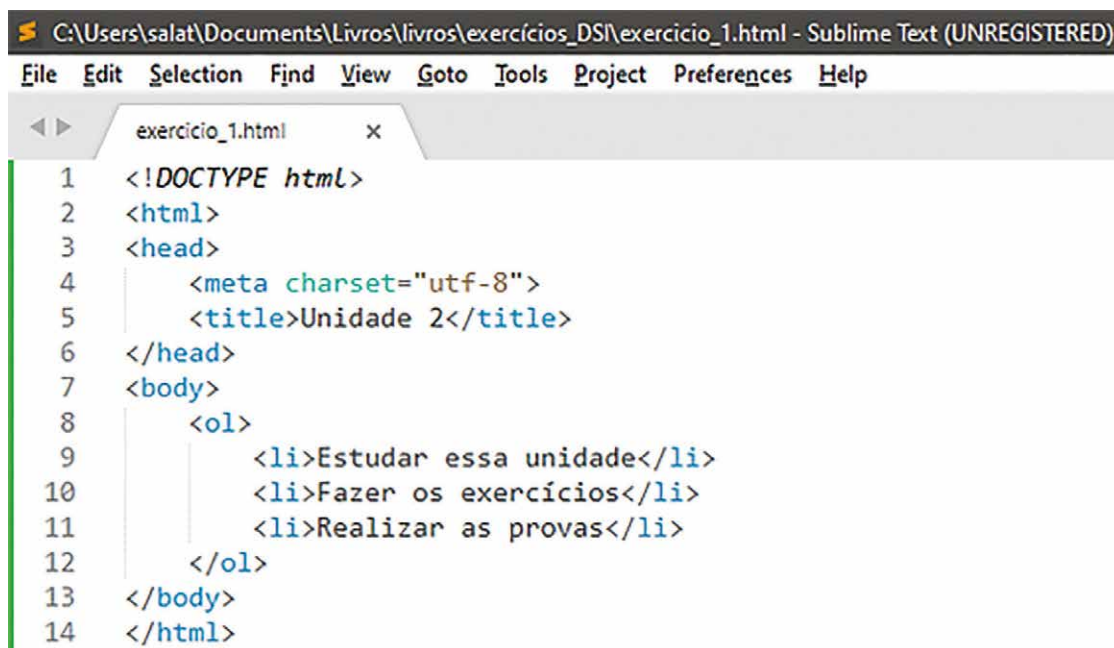
- **None:** os itens da lista ficam sem marcador.

A sintaxe da tag de abertura <ul> com o atributo type é a seguinte:

```
<ul type="disc | circle | square | none">
```

## 4.1.2 Lista ordenada

Uma lista ordenada deve ser criada utilizando o elemento <ol>. Cada item da lista deve ser posicionado dentro das tags <li> e </li>. As listas ordenadas são utilizadas quando a ordem dos itens é importante. Nesse caso, em vez de *bullets*, os itens da lista serão identificados por numeração crescente. Vemos um exemplo de código na figura a seguir.



```
C:\Users\salat\Documents\Livros\livros\exercicios_DSI\exercicio_1.html - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
exercicio_1.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Unidade 2</title>
6 </head>
7 <body>
8     <ol>
9         <li>Estudar essa unidade</li>
10        <li>Fazer os exercícios</li>
11        <li>Realizar as provas</li>
12    </ol>
13 </body>
14 </html>
```

Figura 79 – Exemplo de aplicação de lista ordenada

A execução do código no navegador resulta no que é mostrado na figura a seguir. Note a exibição da numeração no início de cada um dos itens da nossa lista.

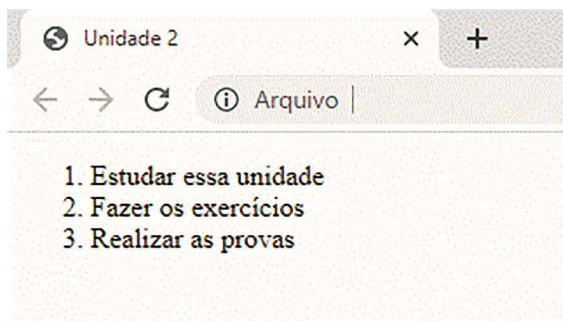


Figura 80 – Página exibida no browser: lista ordenada



A numeração de itens em uma lista ordenada geralmente começa com 1. No entanto, se você deseja alterar isso, pode usar o atributo **start** do elemento `<ol>`, conforme apresentado a seguir.

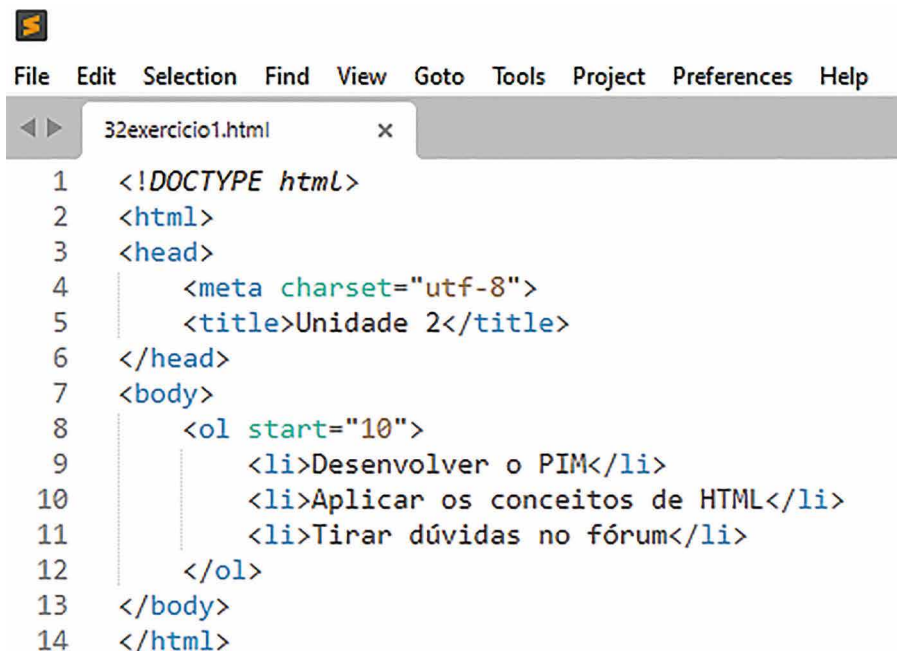


Figura 81 – Exemplo de aplicação de lista ordenada com aplicação do atributo start

Observe na figura seguinte que a numeração da lista, na execução, começará por 10.

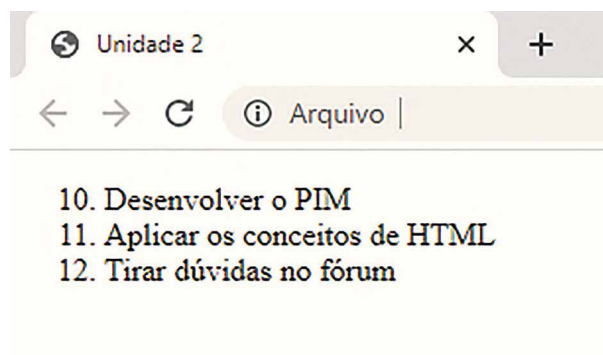


Figura 82 – Página exibida no browser: lista ordenada com atributo start

### 4.1.2.1 Valor do item da lista ordenada `<li value="">`

Você pode definir o valor de um item no meio da lista ordenada manualmente, se não desejar que ele siga a letra ou o número anterior. Basta definir o *value* do atributo `<li>` do item que deseja alterar, conforme apresentado na figura a seguir. Os itens subsequentes seguirão o novo valor.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Unidade 2</title>
6 </head>
7 <body>
8   <ol>
9     <li value="150">Café</li>
10    <li>Leite</li>
11    <li>Suco</li>
12  </ol>
13 </body>
14 </html>

```

Figura 83 – Exemplo de utilização do atributo **value** do elemento `<li>`

## 4.1.3 Lista de descrição

Uma lista de descrição é uma lista de itens que são acompanhados de uma descrição, ou seja, há uma definição de cada item. A lista de descrição é criada usando o elemento `<dl>`, em conjunto com o elemento `<dt>`, que especifica um termo, e com o elemento `<dd>`, que especifica a definição do termo. Os browsers geralmente processam esse tipo de lista colocando os termos e as definições em linhas separadas, posicionando as definições com um recuo da margem.

```

1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="utf-8">
5   <title>Unidade 2</title>
6 </head>
7 <body>
8   <dl>
9     <dt>Unidade 2</dt>
10    <dd>Estudar o conteúdo</dd>
11    <dt>Unidade 3</dt>
12    <dd>Esse conteúdo é interessante!</dd>
13  </dl>
14 </body>
15 </html>

```

Figura 84 – Exemplo de lista de descrição

A execução do código no navegador resulta no que é mostrado a seguir.

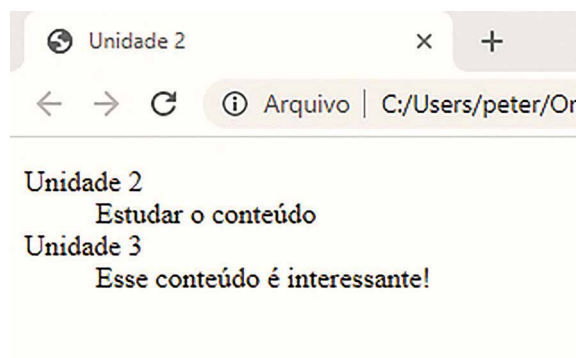


Figura 85 – Página exibida no browser: lista de descrição

### 4.1.4 Atributo type do item de uma lista: <li type="">

O atributo type pode ser aplicado ao elemento <li>, de forma a modificar itens específicos da lista. Com ele, pode-se definir o tipo de numeração ou de marcador para um item no meio da lista. Esse atributo pode ser aplicado tanto às listas ordenadas quanto às listas não ordenadas.

Os valores do atributo type do elemento <li> para listas ordenadas são:

- **1**: valor padrão, utilizado para especificar a numeração.
- **a**: organiza os itens da lista com letras minúsculas.
- **A**: organiza os itens da lista com letras maiúsculas.
- **i**: organiza os itens da lista com algarismos romanos em formato minúsculo.
- **I**: organiza os itens da lista com algarismos romanos em formato maiúsculo.

Observe a sintaxe, com diversos valores possíveis aplicáveis a esse atributo dentro de uma lista ordenada:

```
<li type="1 | a | A | i | I">
```

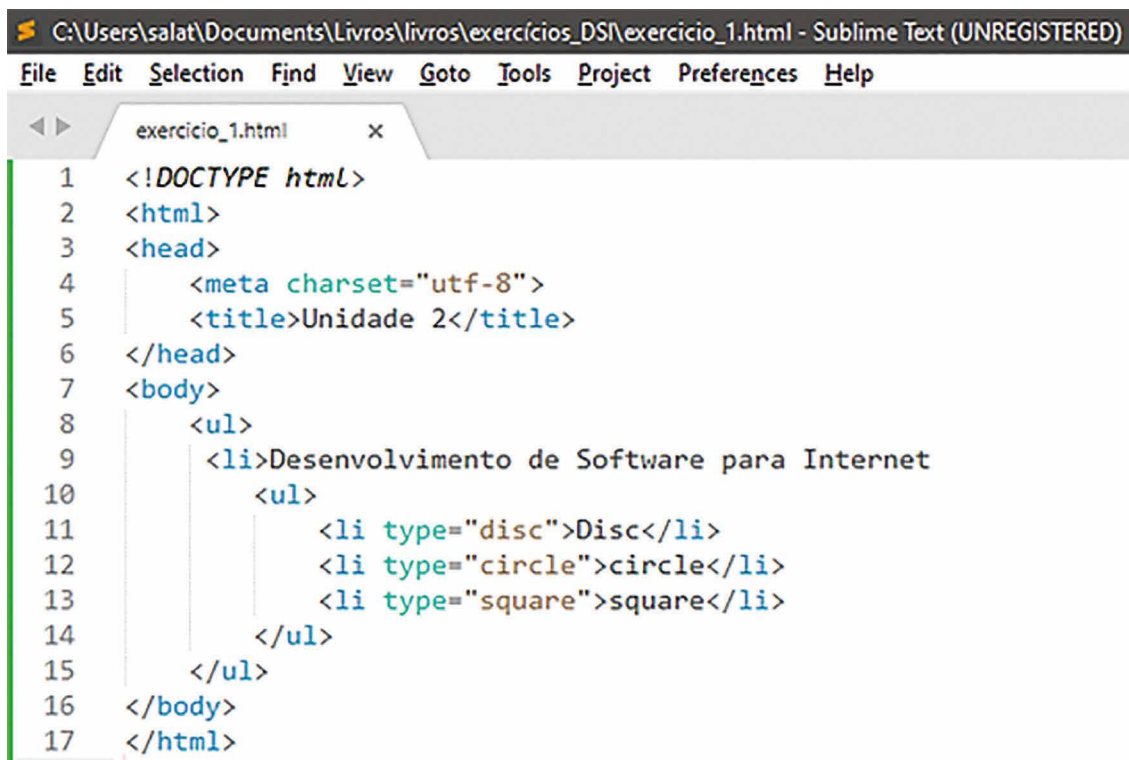
Já os valores do atributo type do elemento <li> para listas não ordenadas são:

- **Disc**: marcador padrão, representando o bullet redondo.
- **Circle**: marcador representado por uma circunferência não preenchida.
- **Square**: marcador quadrado.

Observe a sintaxe, com diversos valores possíveis aplicáveis a esse atributo dentro de uma lista não ordenada:

```
<li type="disc | circle | square">
```

Vejamos um exemplo da utilização do atributo type do elemento <li> em uma lista não ordenada.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4     <meta charset="utf-8">
5     <title>Unidade 2</title>
6 </head>
7 <body>
8     <ul>
9         <li>Desenvolvimento de Software para Internet
10            <ul>
11                <li type="disc">Disc</li>
12                <li type="circle">circle</li>
13                <li type="square">square</li>
14            </ul>
15        </li>
16    </ul>
17 </body>
18 </html>
```

Figura 86 – Exemplo de tipificação de lista



### Observação

O atributo type do elemento <li> não é suportado pelo HTML5.



### Lembrete

A lista não ordenada é usada para criar uma lista de itens relacionados, em nenhuma ordem específica.

A lista ordenada é usada para criar uma lista de itens relacionados, em uma ordem específica.

A lista de descrição é usada para criar uma lista de termos e suas descrições.

### 4.2 Introdução ao XHTML

Você já deve ter notado que algumas páginas web apresentam a extensão .xhtml em seus arquivos. Vamos, aqui, aprender um pouco a respeito desse tema.

HTML e XHTML são duas linguagens nas quais as páginas da web podem ser criadas, sendo que o HTML tem como base de concepção o SGML (*standard generalized markup language*). Em contrapartida, o XHTML tem como base de concepção o XML (*extensible markup language*). O XML pode ser entendido como uma família de linguagens de marcação, que define um conjunto de regras para codificar documentos. O XML foi agregado ao HTML com o propósito de facilitar o compartilhamento de informações pela internet.

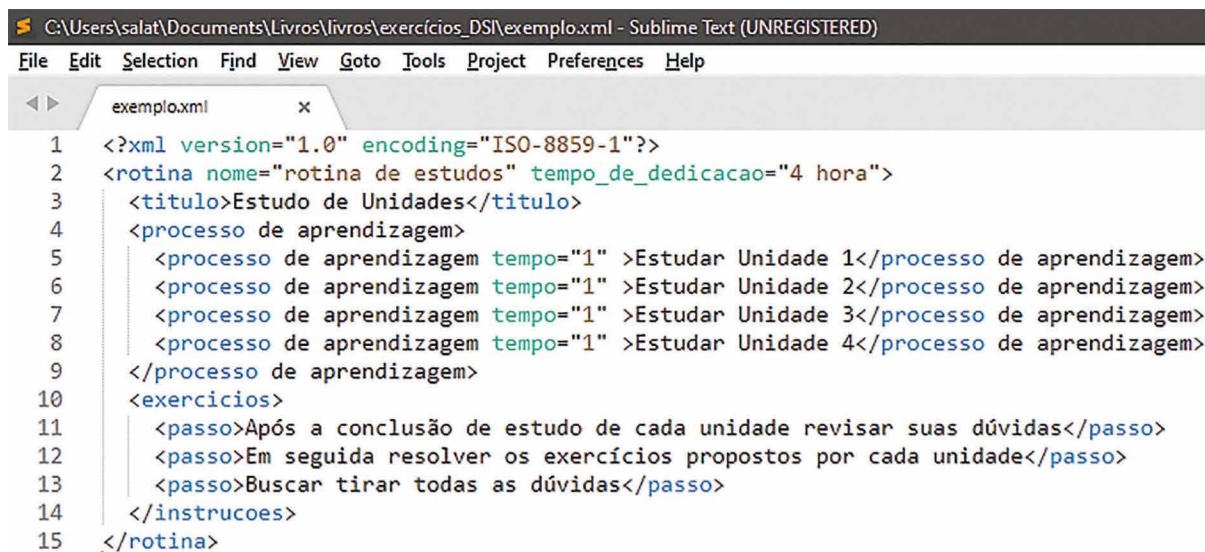
Veja, a seguir, um exemplo de código SGML.

```
<QUOTE TYPE="SGML">
```

Desenvolvimento de software para <ITALICS>internet</ITALICS><BR>.

```
</QUOTE>
```

Agora, veja um exemplo de código XML.



```
C:\Users\salat\Documents\Livros\livros\exercicios_DSI\exemplo.xml - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help

exemplo.xml x
1 <?xml version="1.0" encoding="ISO-8859-1"?>
2 <rotina nome="rotina de estudos" tempo_de_dedicacao="4 hora">
3   <titulo>Estudo de Unidades</titulo>
4   <processo de aprendizagem>
5     <processo de aprendizagem tempo="1" >Estudar Unidade 1</processo de aprendizagem>
6     <processo de aprendizagem tempo="1" >Estudar Unidade 2</processo de aprendizagem>
7     <processo de aprendizagem tempo="1" >Estudar Unidade 3</processo de aprendizagem>
8     <processo de aprendizagem tempo="1" >Estudar Unidade 4</processo de aprendizagem>
9   </processo de aprendizagem>
10  <exercicios>
11    <passo>Após a conclusão de estudo de cada unidade revisar suas dúvidas</passo>
12    <passo>Em seguida resolver os exercícios propostos por cada unidade</passo>
13    <passo>Buscar tirar todas as dúvidas</passo>
14  </instrucoes>
15 </rotina>
```

Figura 87 – Exemplo de arquivo XML



## Saiba mais

Para agregar conhecimento aos estudos de HTML e XHTML, leia o artigo indicado a seguir.

ISHIDA, R. Servindo HTML & XHTML. Tradução: Maurício Samy Silva. W3C, 2016. Disponível em: <https://bit.ly/3Lpj57>. Acesso em: 5 maio 2022.

No contexto histórico, o XHTML foi uma derivação do HTML4 para estar em conformidade com os padrões de XML. Portanto, o XHTML é extremamente alinhado e padronizado quando colocado em comparação ao HTML e não possibilita ao programador escapar de falhas de codificação e estrutura no código. Em outras palavras, o XHTML pode ser entendido como uma extensão do HTML4, funcionando de forma similar ao HTML, mas seguindo uma estrutura cujas regras são mais rígidas, baseadas em XML.

Embora o HTML seja uma linguagem de marcação de texto muito popular para renderizar páginas web, o HTML4 pode não funcionar tão bem em alguns dispositivos, como televisores ou celulares. Assim, o XHTML foi concebido objetivando a melhoria da exibição das páginas web em diversos dispositivos, além da melhoria da acessibilidade do conteúdo.

O XML exige que todos os elementos tenham marca de abertura e de fechamento e estejam aninhados adequadamente. Consequentemente, o XHTML também exigirá isso do código de suas páginas web. Por exemplo, `<br>` é uma tag válida em HTML, mas é necessário escrever `<br />` em XHTML, já que essa sintaxe corresponde à abertura e ao fechamento do elemento na mesma marca (o que é aceitável em XHTML). O quadro a seguir mostra uma comparação entre as duas linguagens de marcação.

**Quadro 9 – Comparação HTML x XHTML**

HTML	XHTML
A linguagem de marcação HTML ou hypertext é considerada, de forma mundial, a principal linguagem de marcação para efetuar a criação de páginas da web e outras informações que podem proporcionar a exibição em um browser	A linguagem de marcação XHTML ( <i>extensible hypertext markup language</i> ) é considerada da família de linguagens de marcação XML, que tem por finalidade o espelhamento ou mesmo a extensão de versões utilizadas em <i>hypertext markup language</i> (HTML), linguagem na qual as páginas de atuação da web são gravadas
html, .htm	.xhtml, .xht, .xml, .html, .htm
text / html	application / xhtml + xml
W3C & WHATWG	Consórcio na World Wide Web
Formato de arquivo do documento	Linguagem de marcação
SGML	XML, HTML
As páginas da web são escritas em HTML	A versão estendida do HTML é mais rígida e baseada em XML
Estrutura flexível que requer analisador específico de HTML branda	Subconjunto restritivo de XML; precisa ser analisado com analisadores XML padrão
Proposta por Tim Berners-Lee em 1987	Recomendação do World Wide Web Consortium em 2000



### 4.2.1 Recursos de documentos HTML e XHTML

Os documentos HTML têm como composição a base de elementos com três componentes:

- um par de tags de elemento (de início e de término);
- atributos de elemento disponibilizados nas tags;
- conteúdo real, textual e gráfico, em que um elemento HTML é considerado tudo o que existe entre as tags.

Os documentos XHTML têm por finalidade apenas a definição de um elemento raiz, ou seja, todos os elementos, incluindo variáveis, devem estar por definição em letras minúsculas, e os valores atribuídos devem necessariamente estar entre aspas, fechados e aninhados para serem devidamente reconhecidos. Esse é um requisito obrigatório em XHTML, que, diferentemente do HTML, é tratado de forma opcional, de maneira que a declaração do doctype delimitaria e colocaria regras a serem seguidas pelos documentos que estão desenvolvidos.

Além das diferenças das declarações de abertura de um documento que existe entre um documento HTML 4.01 e XHTML 1.0, cada uma das DTDs (*document type definition*) correspondentes são tratadas de forma gradativa e sintática. A sintaxe subjacente do HTML possibilita diversos atalhos que o XHTML não possibilita, como elementos com tags de abertura ou de fechamento opcionais ou até mesmo o propósito de trabalho de elementos vazios que não devem ter como premissa uma tag final.

O XHTML exige que todos os elementos possibilitem uma definição de abertura e uma definição de fechamento, mas também permite a introdução de um novo atalho: uma tag XHTML pode ser aberta e fechada dentro da mesma tag, incluindo, para fins de desenvolvimento, uma barra antes do final da tag; por exemplo: `<br/>`. A inserção dessa abreviação, que não é utilizada na declaração do SGML para HTML 4.01, pode gerar um cenário de confusão entre os softwares anteriores não familiarizados com a nova convenção estabelecida. Uma correção seria a inclusão de um espaço antes de fechar a tag; por exemplo: `<br />`.

HTML e XHTML podem ser considerados diretamente relacionados e, portanto, podem ser documentados juntos: o HTML 4.01 e o XHTML 1.0 possibilitam três subespecificações definidas, como estrito, flexível e conjunto de quadros, considerando que a diferença de inicialização de declarações para um documento distingue HTML e XHTML.

O HTML possibilita também atalhos como elementos com tags opcionais, elementos vazios sem tags finais. Já o XHTML é muito rigoroso com relação à inicialização e à finalização de tags.

O XHTML utiliza o atributo de funcionalidade, que tem por definição a linguagem incorporada, ou seja, todos os requisitos de sintaxe do XML estão incluídos diretamente em um documento XHTML bem formado.



### Observação

MIME é a sigla em inglês que corresponde a *multipurpose internet mail extensions*, sendo um padrão utilizado inicialmente para indicar como o conteúdo de uma mensagem de e-mail deve ser tratado, trazendo informações como codificação utilizada, formato do dado, entre outras. Com essas informações, o software que irá interpretá-lo consegue tratá-lo de forma adequada. Hoje, o tipo MIME é um identificador padrão utilizado na internet para indicar o tipo de dado que um arquivo transmitido pela rede contém. Dessa forma, browsers costumam usar o tipo MIME para determinar qual ação usar como padrão quando um recurso é obtido. Documentos de extensão .html possuem tipo MIME text/html e devem ser tratados como HTML clássico. Documentos de extensão .xhtml possuem tipo MIME application/xhtml+xml.

Um documento XHTML atrelado a um tipo MIME text/html deve ser analisado e interpretado de forma obrigatória como HTML clássico. Assim, uma folha de estilo escrita para um documento XHTML que está sendo atrelado a um tipo MIME text/html pode não funcionar conforme o esperado. Esse problema não ocorreria se o documento fosse atrelado a um tipo MIME application/xhtml+xml.

Para que possamos, de maneira objetiva, entender as pequenas diferenças entre HTML e XHTML, podemos considerar a transformação de um documento XHTML 1.0 válido e bem formatado em um documento HTML 4.01 válido. Para a realização dessa tradução, precisamos executar alguns passos, como os citados a seguir.

- Observar que a linguagem (idioma) para definição de um elemento deve ser especificada com um atributo **lang**, e não com o atributo XHTML xml:lang. Ou seja, o XHTML utiliza o atributo de funcionalidade, que tem por definição a linguagem do XML.
- Remover o espaço para nome XML (xmlns=URI): o HTML não possui recursos para namespaces.



### Observação

URI, do inglês, *uniform resource identifier*, tem como definição uma sequência de caracteres que possibilita a identificação de um recurso específico, ou seja, para garantir sua granularidade, todos os URIs seguem um conjunto predefinido de regras determinadas de sintaxe, que consistem na sua extensão em um meio de esquema de nomeação hierárquico determinado de forma separada; por exemplo: http://.

Essa identificação tem por finalidade possibilitar a interação com representações do recurso em uma rede, baseada em vias da World Wide Web, utilizando, assim, protocolos específicos.



Veja um exemplo de aplicação URI.

```
https://exemplounip.com/caminho/unidade.txt#fragmento
//exemplounip.com/caminho/unidade.txt
/caminho/unidade.txt
caminho /unidade.txt
/caminho/unidade.txt
```

Efetua-se o ajuste da declaração do tipo de documento de XHTML 1.0 para HTML 4.01. Se atribuído, remover a declaração XML: `<?xml version="1.0" encoding="utf-8"?>`.

Deve-se também verificar no arquivo se o tipo MIME do documento está definido como `text/html`. No entanto, para HTML e XHTML, isso vem por definição do Content-Type, cabeçalho HTTP enviado previamente pelo servidor.

Por fim, deve-se ajustar a sintaxe do elemento vazio XML para um elemento vazio do estilo HTML (`<br />` para `<br>`).

### 4.2.2 Estrutura de um documento XHTML

Na estrutura do documento XHTML, temos o que segue.

- Doctype XHTML é obrigatório.
- O atributo `xmlns` em `<html>` é obrigatório.
- `<html>`, `<head>`, `<title>` e `<body>` são obrigatórios.

Os elementos XHTML devem:

- estar adequadamente aninhados;
- sempre estar fechados;
- estar em minúsculas;
- ter um elemento raiz.

Nos atributos XHTML, temos o que segue.

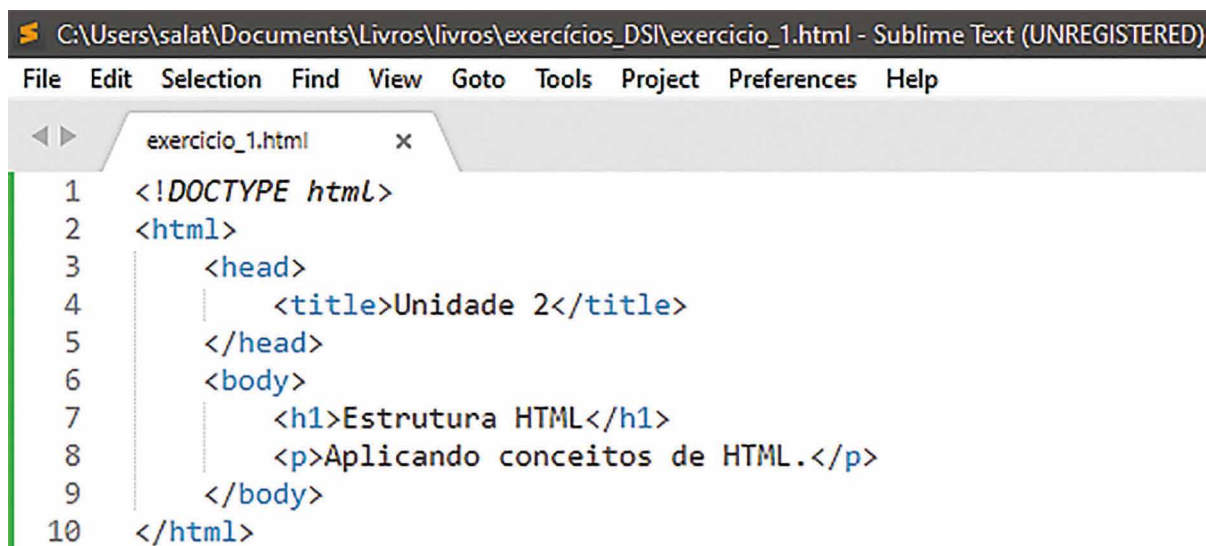
- Os nomes dos atributos devem estar em minúscula.
- Os valores dos atributos devem ser citados.
- A minimização de atributos é proibida.

## 4.2.3 Como converter HTML para XHTML

A conversão de um documento que está em HTML para XHTML acontece por meio dos passos mostrados a seguir.

- Adicionar um XHTML `<!DOCTYPE>` ao início de cada página.
- Adicionar um atributo `xmlns` ao elemento `<html>` de cada página.
- Alterar todos os nomes de elementos para minúsculas.
- Fechar todos os elementos vazios.
- Alterar todos os nomes de atributo para minúsculas.
- Citar todos os valores de atributo.

Veja um exemplo de marcação em HTML.

A screenshot of a Sublime Text editor window. The title bar shows the file path 'C:\Users\salat\Documents\Livros\livros\exercícios\_DSI\exercicio\_1.html - Sublime Text (UNREGISTERED)'. The menu bar includes 'File', 'Edit', 'Selection', 'Find', 'View', 'Goto', 'Tools', 'Project', 'Preferences', and 'Help'. The editor has a single tab titled 'exercicio\_1.html'. The code is as follows:

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Unidade 2</title>
5     </head>
6     <body>
7         <h1>Estrutura HTML</h1>
8         <p>Aplicando conceitos de HTML.</p>
9     </body>
10 </html>
```

Figura 88 – Exemplo de aplicação da estrutura HTML

Agora, observe a conversão da estrutura do exemplo anterior em XHTML.



```
1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
3
4 <html xmlns="http://www.w3.org/1999/xhtml">
5
6 <head>
7 <title>Unidade 2</title>
8 </head>
9
10 <body>
11 Estudo de XHTML
12 </body>
13
14 </html>
```

Figura 89 – Exemplo de aplicação da estrutura XHTML

Note que as linhas 1 e 2 do código contêm um doctype específico do XHTML:

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

Na linha 4, vemos o atributo `xmlns` adicionado à marca de início da tag `<html>`. O valor desse atributo (a URL que aparece entre as aspas) é padrão, não devendo ser modificado, pois ele indica que a sintaxe utilizada no documento é XHTML:

```
<html xmlns="http://www.w3.org/1999/xhtml">
```

Note, também, que todas as tags foram escritas em letras minúsculas (não poderíamos, por exemplo, escrever `<BODY>`, que seria aceitável em HTML). Além disso, todos os elementos do exemplo possuem tags de abertura e de fechamento.



### Observação

Nos dias de hoje, o XHTML ainda é utilizado, mas não costuma ser a linguagem de marcação preferida dos desenvolvedores web. Ela foi desenvolvida para ser uma versão aprimorada do HTML4 por meio da incorporação de recursos XML, porém o próprio HTML5 é a linguagem de marcação mais popular atualmente. Isso ocorre porque o HTML5 conta com a adição de muitos recursos em relação a suas versões anteriores, que garantem a compatibilidade buscada pelo XHTML, mas sem imposições tão restritas de sintaxe.



### Resumo

Na unidade II, vimos como podemos combinar e integrar técnicas de desenvolvimento web e conhecemos tags que auxiliam na validação das informações inseridas em uma página web.

Abordamos conceitos de implementação que foram aplicados em HTML5 a fim de que, com isso, aumentemos a consistência do trabalho executado na web. Também conseguimos trabalhar com formulários, que continuam sendo de extrema importância para o cadastro de informações básicas de um cliente, como listas ordenadas e listas não ordenadas, e com tags que ajudam na construção do formulário.

Nesta unidade, estudamos métodos que auxiliam no funcionamento de um formulário, como os métodos get e post, utilizados em cenários específicos para o desenvolvimento web.

Vale ressaltar que existem inúmeras tags que poderiam agregar muito no desenvolvimento web e no enriquecimento do layout, tornando as páginas dinâmicas e agradáveis.

Assim, frisamos que toda construção de páginas web com HTML5 tem como focos principais a atuação em desenvolvimentos de aplicações web com agilidade e qualidade e a intenção de proporcionar a visualização dessas páginas em qualquer dispositivo conectado à internet.

Adicionalmente, destacamos que o aprendizado adquirido será de grande importância para a aplicação dos conceitos aqui vistos no dia a dia do desenvolvimento de software para internet.



## Exercícios

**Questão 1.** (Ibade 2018, adaptada) Um analista de sistemas está elaborando uma página, para um site na internet, que inclui um formulário codificado em HTML. O trecho do código correspondente ao formulário HTML é mostrado a seguir.

```
<form>
  <div align="left"><h1><font face="Arial" size="4">
    <b>PARAÍBA 2018</b></font></h1>
  </div><p><b><font face="Arial" size="4">Nome:
  <input type="text" nome="Nome" size="40"><br>Cidade:
  <input type="checkbox" name="CIDADE"
    value="c1" checked>JOÃO PESSOA
  <input type="checkbox" name="CIDADE"
    value="c2">CAMPINA GRANDE</font></b></p>
</form>
```

Figura 90

Após a execução, o formulário gerado pelo código no navegador é:

A) **PARAÍBA 2018**  
 Nome:   
 Cidade: ☐ JOÃO PESSOA ☒ CAMPINA GRANDE

B) **PARAÍBA 2018**  
 Nome:   
 Cidade: ☒ JOÃO PESSOA ☐ CAMPINA GRANDE

C) **PARAÍBA 2018**  
 Nome:   
 Cidade: ☒ CAMPINA GRANDE ☐ JOÃO PESSOA

D) **PARAÍBA 2018**  
 Nome:   
 Cidade: ☒ CAMPINA GRANDE ☐ JOÃO PESSOA

E) **PARAÍBA 2018**  
 Nome:   
 Cidade: ☒ JOÃO PESSOA ☐ CAMPINA GRANDE

Resposta correta: alternativa B.

## Análise da questão

Vamos fazer a análise do código apresentado no enunciado. O formulário começa exibindo um título <h1>, inserido dentro de uma divisão, onde se lê PARAÍBA 2018. Depois, é criado um campo de entrada de dados <input>, do tipo texto, destinado à inserção do nome. Até esse ponto do código, esperaríamos a exibição demonstrada a seguir.

### PARAÍBA 2018

Nome:

Em sequência, é criado um novo elemento <input>, do tipo checkbox, com nome "CIDADE". O texto apresentado ao usuário é JOÃO PESSOA. O atributo checked desse elemento indica que a caixa desse checkbox deve ser exibida previamente e selecionada no navegador. Até esse ponto, teríamos a exibição:

### PARAÍBA 2018

Nome:   
Cidade: ☒ JOÃO PESSOA

O terceiro e último elemento <input> do código também é do tipo checkbox, com nome "CIDADE". O texto apresentado ao usuário é CAMPINA GRANDE. Por ter o mesmo nome do checkbox anterior, ambos participam do mesmo grupo de caixas de seleção. Note que não há atributo checked visível no código. Nesse caso, o elemento será exibido, por padrão, com sua caixa não selecionada, conforme visto a seguir.

### PARAÍBA 2018

Nome:   
Cidade: ☒ JOÃO PESSOA ☐ CAMPINA GRANDE

**Questão 2.** Leia o texto a seguir a respeito da evolução da linguagem HTML ao longo dos anos.

A versão 4 da HTML tornou-se uma recomendação World Wide Web Consortium (W3C) em 1998, para atender às necessidades da web estática, a Web 1.0. Nesse período, era utilizada apenas para postar conteúdo sem que o usuário pudesse interagir com ele. Desde então, a web evoluiu, agregando novas tecnologias e conceitos que tornaram possível a publicação de conteúdo dinâmico (Web 2.0), dando poder ao usuário para interagir com os websites.

Apesar de os recursos da HTML versão 4 ainda serem a referência para o desenvolvimento de aplicações web, essa versão não provê recursos suficientemente eficientes para que as diversas tecnologias necessárias ao desenvolvimento de aplicações web possam interoperar de maneira adequada.

Ao perceber que a HTML versão 4 não atendia adequadamente às necessidades do desenvolvimento para Web 2.0, a W3C criou o XHTML, uma espécie de HTML com as regras rigorosas da linguagem XML. O XHTML também não agradou a comunidade de desenvolvedores por não atender a seus anseios de mudança. Então, algumas empresas e desenvolvedores criaram a Web Hypertext Application Technology Working Group (WHATWG), uma organização com foco no trabalho com formulários web e aplicações.

Tempos depois, em 2006, a W3C fez uma parceria com a WHATWG e, juntas, criaram a linguagem HTML versão 5 para atender tanto às normas e aos padrões da W3C quanto aos anseios da comunidade de desenvolvedores. A HTML5 é o resultado da quinta maior revisão da linguagem HTML, considerada o novo padrão para Web, que une o que há de melhor na HTML versão 4, XHTML e HTML DOM, e adicionará um conjunto de novos recursos.

TERUEL, E. C. *HTML5: guia prático*. 2. ed. São Paulo: Érica, 2014. p. 17.

Mesmo com a criação do HTML5, a linguagem XHTML ainda é muito presente em documentos web e, por isso, devemos nos familiarizar com seus padrões. A respeito das linguagens HTML e XHTML, avalie as afirmativas.

I – A instrução doctype do HTML5 é mais simples do que a instrução das versões HTML 4.01 ou XHTML 1.0.

II – Em documentos escritos com a linguagem XHTML, os valores dos atributos devem vir entre aspas e não devem ser vazios.

III – Na estrutura dos elementos da versão HTML5, todas as marcas de abertura devem ser acompanhadas de suas respectivas marcas de fechamento. Quando não possuírem fechamento, os caracteres `</>` deverão ser utilizados.

É correto o que se afirma em:

A) I, apenas.

B) II, apenas.

C) I e II, apenas.

D) II e III, apenas.

E) I, II e III.

Resposta correta: alternativa C.

Justificativa: na versão HTML5, a declaração do doctype é feita apenas com o trecho `<!DOCTYPE html>`. A declaração dessa instrução é um pouco mais complexa nas outras versões. Veja exemplos a seguir.

- ```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
```

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

Justificativa: os valores dos atributos em XHTML devem ser citados, obrigatoriamente. Desse modo, eles não podem ser vazios. Além disso, os valores são posicionados, necessariamente, entre aspas.

Justificativa: em um documento HTML5, não é obrigatório que todos os elementos sejam estruturados com tags de abertura e fechamento. Essa é uma imposição da linguagem XHTML.

[illegible]