

1 Introdução

Neste material, prosseguimos com nossos estudos sobre HTML.

2 Desenvolvimento

(Ambiente) O primeiro passo é obter um ambiente para desenvolvermos nossos testes. Há diversos ambientes on-line de boa qualidade que nos permitem desenvolver sem ter de instalar um ambiente local. Um deles é o Replit:

<https://replit.com/>

Um outro muito bom é o CodePen:

<https://codepen.io/>

Há também o Visual Studio Code for the Web:

<https://vscode.dev/>

Você pode testá-los e adotar aquele que achar mais interessante.

Há também a possibilidade de utilizar um ambiente local. Esta é a abordagem adotada neste material. O editor de texto/código que utilizamos é o Visual Studio Code:

<https://code.visualstudio.com/>

(Uma pasta no seu sistema de arquivos) Comece criando uma pasta no seu sistema de arquivos. Se estiver usando o Windows, uma boa opção pode ser

C:\Users\seuusuario\Documents\html

Depois disso, abra o VS Code e clique em **File >> Open Folder**. Navegue até a pasta que acabou de criar para vincular o VS Code a ela.

(Forms) Quando desenvolvemos uma página HTML, muitas vezes ela possui **elementos** que permitem ao usuário digitar dados que serão enviados a um servidor. Para que esse **envio** seja possível, utilizamos um **elemento** do tipo **form**. Os **filhos** de um **form** podem ser de diferentes tipos. Eles podem permitir que o usuário:

- digite um texto
- escolha um item em um menu
- escolha uma cor
- escolha um número
- escolha uma data

Entre muitos outros. Cada **elemento** filho de um **form** que **permite** ao usuário realizar a entrada de algum tipo de dado recebe o nome de **controle**.

(Novo arquivo) Para este exemplo, comece criando um arquivo chamado **primeiro_form.html**. Veja seu conteúdo inicial:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>

  </body>

</html>
```

Veja como criamos um form. O nome do elemento é form mesmo:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>

    <form>

  </form>

  </body>
</html>
```

Nota. Em geral, o navegador não exibirá coisa alguma por conta da existência de um elemento form no documento HTML. A razão de ser de um elemento form é simplesmente agrupar elementos capazes de receber dados do usuário e viabilizar seu envio - feito pelo navegador - a um servidor.

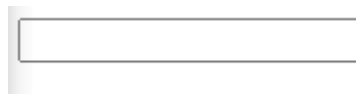
(Elemento input) Dizemos que os controles de um form permitem que o usuário faça a **entrada de dados**. Há um elemento HTML chamado **input** próprio para este fim. Ele possui um atributo chamado **type** que permite variar o tipo dos valores que o usuário poderá informar. Veja como obter texto do usuário:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>

    <form>
      <input type="text">
    </form>

  </body>
</html>
```

Veja o resultado esperado:



(Rótulos) Observe que já é possível digitar no campo. Entretanto, o usuário não sabe a que se refere esse campo. Precisamos associar um rótulo a ele. Podemos fazê-lo utilizando um elemento HTML chamado **label**. Digamos que o campo serve para que o usuário digite o seu primeiro nome. Neste caso, vamos criar um label da seguinte forma:

- Seu corpo conterá o texto que desejamos como rótulo
- Seu atributo **for** terá um valor associado, que indicará o vínculo entre o label e o input a que se refere.
- O valor associado ao atributo **for** do label será o id do input, finalizando assim o vínculo entre eles.

Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>

    <form>
      <label
        for="primeiroNome">
        Primeiro nome:
      </label>
      <input id="primeiroNome" type="text">
    </form>

  </body>
</html>
```

Veja o resultado esperado no navegador:

Primeiro nome:

Observe que se você clicar sobre o texto “Primeiro nome:” no navegador, o cursor será levado automaticamente ao elemento input a que o label está associado. Entretanto, ainda mais importante que isso, é a semântica do código. Fica claro para quem lê o código (incluindo leitores de tela) que esse rótulo está associado a esse input.

Veja um novo campo em que o usuário pode digitar seu sobrenome:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label
        for="primeiroNome">
        Primeiro nome:
      </label>
      <input id="primeiroNome" type="text">
      <label for="sobrenome">Sobrenome:</label>
      <input id="sobrenome" type="text">
    </form>
  </body>
</html>
```

Resultado esperado:

Primeiro nome: Sobrenome:

(Associação entre label e input sem id) A associação entre label e input pode ser feita sem a utilização de id. Basta aninhar o elemento input dentro do rótulo a que desejamos associá-lo. Observe:

```

<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label>
        Primeiro nome:
        <input id="primeiroNome" type="text">
      </label>
      <label>
        Sobrenome:
        <input id="sobrenome" type="text">
      </label>
    </form>
  </body>
</html>

```

O resultado visual e a semântica que o código apresenta permanecem os mesmos.

(Envio de um form) Uma vez que tenha preenchido todos os campos, o usuário desejará enviar o form. Isso pode ser feito de diferentes formas. Uma delas envolve o uso de um elemento do tipo **input** com atributo **type** associado ao valor **submit**. O atributo **value** fica associado ao valor a ser exibido como texto do botão. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label>
        Primeiro nome:
        <input id="primeiroNome" type="text">
      </label>
      <label>
        Sobrenome:
        <input id="sobrenome" type="text">
      </label>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

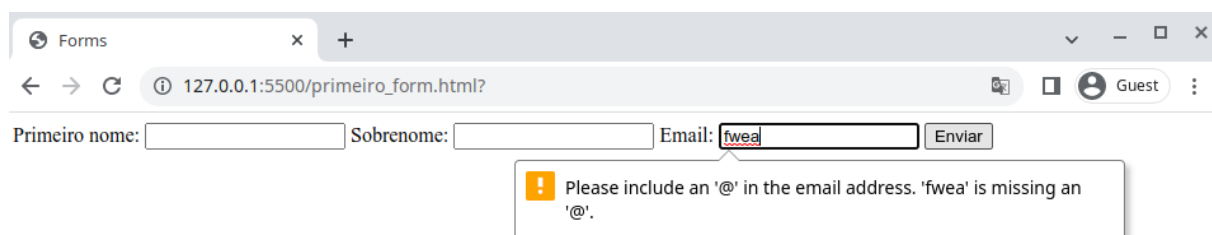
Observe que um `input` com `type` igual a `submit` é apresentado ao usuário como um botão:

Primeiro nome: Sobrenome:

(Outros tipos de input: email) Podemos associar diferentes valores ao atributo `type` de um elemento `input`. Um exemplo é o valor **"email"**. Veja:


```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label>
        Primeiro nome:
        <input id="primeiroNome" type="text">
      </label>
      <label>
        Sobrenome:
        <input id="sobrenome" type="text">
      </label>
      <label for="email">Email:</label>
      <input id="email" type="email">
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

No navegador, digite algo que não representa um e-mail válido e obtenha um resultado parecido com esse:



The screenshot shows a web browser window with the title 'Forms'. The address bar shows the URL '127.0.0.1:5500/primeiro_form.html?'. The page content includes three input fields: 'Primeiro nome:', 'Sobrenome:', and 'Email:'. The 'Email' field contains the text 'fwear'. Below the 'Email' field, a validation error message is displayed: 'Please include an '@' in the email address. 'fwear' is missing an '@'.' The message is enclosed in a box with a yellow warning icon.

Além da validação, caso o usuário acesse essa página usando um dispositivo móvel, o teclado virtual pode aparecer de forma personalizada, com uma tecla reservada para o símbolo @, por exemplo.

(Outros tipos de input: telefone) Um campo apropriado para um número de telefone pode ser representado por um input de type igual a **tel**. Veja:

```
!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label>
        Primeiro nome:
        <input id="primeiroNome" type="text">
      </label>
      <label>
        Sobrenome:
        <input id="sobrenome" type="text">
      </label>

      <label for="email">Email:</label>
      <input id="email" type="email">

      <label for="telefone">Telefone:</label>
      <input id="telefone" type="tel">
```

```

        <input type="submit" value="Enviar">
    </form>
</body>
</html>

```

A exibição feita pelo navegador provavelmente não terá nada de especial. Entretanto, caso o usuário acesse o site com seu dispositivo móvel, o teclado virtual também poderá ser exibido de forma personalizada, talvez dando foco para valores numéricos.

(Outros tipos de input: cor) Veja um input apropriado para a coleta de uma cor escolhida pelo usuário:

```

<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label>
        Primeiro nome:
        <input id="primeiroNome" type="text">
      </label>
      <label>
        Sobrenome:
        <input id="sobrenome" type="text">
      </label>

      <label for="email">Email:</label>
      <input id="email" type="email">
    </form>
  </body>
</html>

```

```
<label for="telefone">Telefone:</label>
<input id="telefone" type="tel">

<label for="cor">Cor:</label>
<input id="cor" type="color">

<input type="submit" value="Enviar">
</form>
</body>
</html>
```

Veja como o navegador renderiza esse controle. Clique sobre a cor para escolher a que desejar.

Primeiro nome: Sobrenome: Email: Telefone:
Cor:

(Outros tipos de input: arquivo) Talvez seja necessário permitir que o usuário escolha um arquivo para fazer upload. Para isso, associamos o valor **file** ao atributo type de um elemento input. Observe:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label>
        Primeiro nome:
        <input id="primeiroNome" type="text">
      </label>
      <label>
        Sobrenome:
        <input id="sobrenome" type="text">
      </label>

      <label for="email">Email:</label>
      <input id="email" type="email">

      <label for="telefone">Telefone:</label>
      <input id="telefone" type="tel">

      <label for="cor">Cor:</label>
      <input id="cor" type="color">

      <label for="arquivo">Arquivo:</label>
      <input id="arquivo" type="file">
```

```

    <input type="submit" value="Enviar">
  </form>
</body>
</html>

```

Há uma lista muito grande de valores que podemos associar ao atributo type do elemento `input`. Visite o link a seguir para conhecê-los:

atributo type elemento input

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input>

Não deixe de fazer seus próprios testes!

(Elemento textarea) É comum utilizar um elemento do tipo **textarea** quando desejamos permitir que o usuário digite um texto potencialmente “grande”. Veja como fazê-lo a seguir. Observe que um `textarea` possui os atributos **cols** e **rows**. Eles permitem especificar largura e altura da caixa.

```

<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="UTF-8">
    <title>Forms</title>
  </head>
  <body>
    <form>
      <label>
        Primeiro nome:
        <input id="primeiroNome" type="text">
      </label>
      <label>
        Sobrenome:

```

Elemento textarea :

Permitir que o usuário digite um texto potencialmente grande

atributos:

- cols: largura da caixa
- rows: altura da caixa

```

        <input id="sobrenome" type="text">
    </label>

    <label for="email">Email:</label>
    <input id="email" type="email">

    <label for="telefone">Telefone:</label>
    <input id="telefone" type="tel">

    <label for="cor">Cor:</label>
    <input id="cor" type="color">

    <label for="arquivo">Arquivo:</label>
    <input id="arquivo" type="file">

    <label for="mensagem">Sua mensagem:</label>
    <textarea id="mensagem" cols="35"
rows="10"></textarea>

    <input type="submit" value="Enviar">
</form>
</body>
</html>

```

(Conjunto fixo de opções de escolha única: o elemento `select`) Pode ser o caso de desejarmos que o usuário faça a escolha de um valor dentro de uma coleção de opções previamente definidas, ao invés de deixá-lo digitar um valor abertamente. Para exibir a coleção de opções como um menu, podemos usar um elemento do tipo `select`. Para essa seção, crie um novo arquivo chamado **opcoes.html**. Veja seu conteúdo inicial:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>

  </body>
</html>
```

A criação de um elemento select, que representa um menu de opções, envolve o aninhamento de elementos do tipo option dentro dele, cada qual representando uma opção do menu. Observe:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
    <label for="estado">Estado:</label>
    <select id="estado">
      <option>São Paulo</option>
      <option>Rio de Janeiro</option>
      <option>Minas Gerais</option>
      <option>Espírito Santo</option>
    </select>
  </body>
</html>
```


Veja o resultado esperado:

Estado:

São Paulo

Rio de Janeiro

Minas Gerais

Espírito Santo

(Conjunto de opções de escolha única: o elemento `input` de `type` igual a `radio`) Também podemos apresentar uma coleção de opções ao usuário utilizando o elemento `input` com `type` associado ao valor `radio`. Veja um desses ainda não muito útil:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
    <label for="estado">Estado:</label>
    <select id="estado">
      <option>São Paulo</option>
      <option>Rio de Janeiro</option>
      <option>Minhas Gerais</option>
      <option>Espírito Santo</option>
    </select>

    <input type="radio">
  </body>
</html>
```

Ele deve ser apresentado pelo navegador assim:

Estado: ☒

(Atributo name para agrupar os elementos radio) Observe que se especificarmos diversos elementos assim, eles serão independentes e o usuário poderá marcar todos. Essa não é exatamente a ideia deste elemento. A ideia é que o usuário possa somente selecionar um. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
    <label for="estado">Estado:</label>
    <select id="estado">
      <option>São Paulo</option>
      <option>Rio de Janeiro</option>
      <option>Minhas Gerais</option>
      <option>Espírito Santo</option>
    </select>

    <input type="radio">
    <input type="radio">
    <input type="radio">
    <input type="radio">
    <input type="radio">

  </body>
</html>
```

Observe como é possível selecionar mais de um:

Estado: ☒ ☒ ☐ ☒ ☐

Para explicar ao navegador que esses elementos fazem parte de um **grupo** e que somente um pode ser selecionado, associamos um valor ao atributo **name** deles. Claro, o valor deve ser igual para todos. Observe:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
    <label for="estado">Estado:</label>
    <select id="estado">
      <option>São Paulo</option>
      <option>Rio de Janeiro</option>
      <option>Minhas Gerais</option>
      <option>Espírito Santo</option>
    </select>

    <input type="radio" name="veiculo">
    <input type="radio" name="veiculo">
    <input type="radio" name="veiculo">
    <input type="radio" name="veiculo">
    <input type="radio" name="veiculo">

  </body>
</html>
```

Faça um novo teste no navegador e perceba que agora já não é possível selecionar mais de um elemento.

(Rótulos para os radios) Claro, agora precisamos associar um texto a cada elemento. Um elemento do tipo label desempenha esse papel de maneira muito apropriada. Veja:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
    <label for="estado">Estado:</label>
    <select id="estado">
      <option>São Paulo</option>
      <option>Rio de Janeiro</option>
      <option>Minhas Gerais</option>
      <option>Espírito Santo</option>
    </select>

    <input type="radio" name="veiculo"><label>X1</label>
    <input type="radio" name="veiculo"><label>XC60</label>
    <input type="radio" name="veiculo"><label>Q5</label>
    <input type="radio" name="veiculo"><label>Discovery</label>
    <input type="radio" name="veiculo"><label>Taycan</label>

  </body>
</html>
```

(Associando cada label a seu input) Não podemos esquecer da semântica da página e de, portanto, associar cada label a seu respectivo input:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
```

```

<label for="estado">Estado:</label>
<select id="estado">
  <option>São Paulo</option>
  <option>Rio de Janeiro</option>
  <option>Minas Gerais</option>
  <option>Espírito Santo</option>
</select>

<input
  id="X1"
  type="radio"
  name="veiculo">
<label for="X1">X1</label>
<input
  id="XC60"
  type="radio"
  name="veiculo">
<label for="XC60">XC60</label>
<input
  id="Q5"
  type="radio"
  name="veiculo">
<label for="Q5">Q5</label>
<input
  id="Discovery"
  type="radio"
  name="veiculo">
<label for="Discovery">Discovery</label>
<input
  id="Taycan"
  type="radio"
  name="veiculo">
<label for="Taycan">Taycan</label>
</body>
</html>

```

(Elemento fieldset: agrupamento os elementos semanticamente e especificando uma pergunta) Ainda há a necessidade de agruparmos os elementos usando um elemento HTML

apropriado. Cuidamos da semântica do documento e viabilizamos a especificação de uma potencial pergunta a que o usuário deve responder escolhendo um dos elementos. Para o agrupamento, vamos usar o elemento HTML **fieldset**. Veja parte de sua documentação, que revela seu propósito:

*The **fieldset** element represents a set of form controls optionally grouped under a common name.*

A pergunta será especificada no corpo de um elemento **legend**. Veja parte de sua documentação, que também revela o seu propósito:

*The **legend** element represents a caption for the rest of the contents of the **legend** element's parent **fieldset** element, if any.*

Fica assim:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
    <label for="estado">Estado:</label>
    <select id="estado">
      <option>São Paulo</option>
      <option>Rio de Janeiro</option>
      <option>Minas Gerais</option>
      <option>Espírito Santo</option>
    </select>
    <fieldset>
      <legend>Escolha um veículo</legend>
      <input
        id="X1"
        type="radio"
        name="veiculo">
      <label for="X1">X1</label>
      <input
        id="XC60"
        type="radio"
        name="veiculo">
      <label for="XC60">XC60</label>
      <input
        id="Q5"
```

```

        type="radio"
        name="veiculo">
<label for="Q5">Q5</label>
<input
    id="Discovery"
    type="radio"
    name="veiculo">
<label for="Discovery">Discovery</label>
<input
    id="Taycan"
    type="radio"
    name="veiculo">
<label for="Taycan">Taycan</label>
</fieldset>
</body>

</html>

```

Veja o resultado esperado:

Estado: São Paulo ▼

Escolha um veículo

☐ X1
 ☐ XC60
 ☐ Q5
 ☐ Discovery
 ☒ Taycan

Observe que a renderização de um fieldset pode envolver a especificação de uma borda e que o texto da pergunta pode ser posicionado sobre ela. Isso depende do software cliente utilizado para acessar a página. Evidentemente, um leitor de tela utilizado por um deficiente visual não possui um mecanismo assim. O uso dos elementos apropriados é fundamental para que o dispositivo possa explicar a estrutura da página o mais detalhadamente possível.

(Conjunto de opções de escolha múltipla: o elemento input de type igual a checkbox) Um elemento input com atributo type associado ao valor checkbox viabiliza a escolha de múltiplos valores feita pelo usuário. Veja um exemplo. Tal qual fizemos anteriormente, também faremos uso dos elementos fieldset e legend.

```

<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Opções</title>
  </head>
  <body>
    <label for="estado">Estado:</label>
    <select id="estado">
      ...
    </select>
    <fieldset>
      <legend>Escolha um veículo</legend>
      ...
    </fieldset>

    <fieldset>
      <legend>De quais cores você gosta?</legend>
      <input
        id="vermelho"
        type="checkbox"
        name="cor">
      <label for="vermelho">Vermelho</label>
      <input
        id="amarelo"
        type="checkbox"
        name="cor">
      <label for="amarelo">Amarelo</label>
      <input
        id="preto"
        type="checkbox"
        name="cor">
      <label for="preto">Preto</label>
    </fieldset>
  </body>
</html>

```

(Tabelas de dados: o elemento table) Nesta seção, construiremos uma tabela para a exibição de uma coleção de dados. Para tal, usamos o elemento `table`. Veja um trecho da sua documentação para entender o seu propósito:

The table element represents data with more than one dimension, in the form of a table.

Para esta seção, crie um novo arquivo de exemplo chamado **tabelas.html**. Veja seu código inicial:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Tabelas</title>
  </head>
  <body>

  </body>
</html>
```

O primeiro passo é definir um elemento do tipo **table**:

```
<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Tabelas</title>
  </head>
  <body>
    <table>

    </table>
  </body>
</html>
```

Cada linha de uma tabela é representada por um elemento **tr** (de **table row**). Aninhados dentro de um **tr**, podemos ter diversos elementos do tipo **td** (de **table data**). Cada **td** representa uma coluna da tabela. No exemplo a seguir, estamos exibindo alguns dados de dois alunos de uma instituição.

Elemento table: dados tabulares

tr: table row (linha)

td: table data

th: table header

thead: grupo cabeçalho

tbody: grupo corpo

tfoot: grupo rodapé

```

<!DOCTYPE html>
<html lang="pt-BR">
  <head>
    <meta charset="utf-8">
    <title>Tabelas</title>
  </head>
  <body>
    <table>
      <!-- primeira linha -->
      <tr>
        <td>123456</td>
        <td>João</td>
        <td>Santos</td>
        <td>10</td>
      </tr>
      <!-- segunda linha -->
      <tr>
        <td>447744</td>
        <td>Maria</td>
        <td>Silva</td>
        <td>9</td>
      </tr>
    </table>
  </body>
</html>

```

Veja o resultado esperado:

123456	João	Santos	10
447744	Maria	Silva	9

Observe que o significado de cada coluna não é claro para o usuário. Claro, desejamos adicionar um cabeçalho a ela. Podemos fazê-lo, com uma nova linha. Entretanto, seus filhos serão do tipo **th** (de table header). Veja:

```

...
<body>
  <table>
    <!-- cabeçalho -->
    <tr>
      <th>RA</th>
      <th>Nome</th>
      <th>Sobrenome</th>
      <th>Média</th>
    </tr>
    <!-- primeira linha -->
    <tr>
      <td>123456</td>

```

Resultado esperado:

RA	Nome	Sobrenome	Média
123456	João	Santos	10
447744	Maria	Silva	9

Idealmente, promovemos a legibilidade do documento HTML englobando as informações do cabeçalho da tabela com um elemento do tipo **thead**. Observe:

```

...
<body>
  <table>
    <!-- cabeçalho -->
    <thead>
      <tr>
        <th>RA</th>
        <th>Nome</th>
        <th>Sobrenome</th>
        <th>Média</th>
      </tr>
    </thead>
    <!-- primeira linha -->
  ...

```

Há também um elemento idealmente utilizado para abrigar os dados da tabela, ou seja, seu “corpo” principal. É o **tbody**. Veja:

```

...
<!-- cabeçalho -->
<thead>
  <tr>
    <th>RA</th>
    <th>Nome</th>
    <th>Sobrenome</th>
    <th>Média</th>
  </tr>
</thead>
<tbody>
  <!-- primeira linha -->
  <tr>
    <td>123456</td>
    <td>João</td>
    <td>Santos</td>
    <td>10</td>
  </tr>
  <!-- segunda linha -->
  <tr>
    <td>447744</td>
    <td>Maria</td>
    <td>Silva</td>
    <td>9</td>
  </tr>
</tbody>
</table>
...

```

Repare que o uso dos elementos `thead` e `tbody` não necessariamente causa impacto visual:

RA	Nome	Sobrenome	Média
123456	João	Santos	10
447744	Maria	Silva	9

Entretanto, seu uso é fortemente recomendado por promover a legibilidade do documento HTML.

Tabelas também podem ter um rodapé. Ele é representado por um elemento HTML `tfoot`. Veja um exemplo:

```
...
<table>
  <!-- cabeçalho -->
  <thead>
    <tr>
      <th>RA</th>
      <th>Nome</th>
      <th>Sobrenome</th>
      <th>Média</th>
    </tr>
  </thead>
  <tbody>
    <!-- primeira linha -->
    <tr>
      <td>123456</td>
      <td>João</td>
      <td>Santos</td>
      <td>10</td>
    </tr>
    <!-- segunda linha -->
    <tr>
      <td>447744</td>
      <td>Maria</td>
      <td>Silva</td>
      <td>9</td>
    </tr>
  </tbody>
  <tfoot>
    <tr>
      <td>--</td>
      <td>--</td>
      <td>--</td>
      <td>9.5</td>
    </tr>
  </tfoot>
</table>
...
```

(Tableless) Observe que seria possível utilizar um elemento HTML table para organizar o leiaute de uma página. Ou seja, decidir se elementos devem ser exibidos lado a lado ou um abaixo do outro, por exemplo.

Entretanto, elementos table somente devem ser utilizados para a exibição de dados. A definição do leiaute de uma página deve ser feita com CSS, que veremos posteriormente.

Tableless (que passa uma ideia parecida com “não use tabelas”) é o nome costumeiramente associado ao método de Web Design que sugere que tabelas não devem ser utilizadas na definição do leiaute de páginas HTML.

Resumindo:

- Use elementos table normalmente, desde que seja com a finalidade de exibir dados.
- Não use elementos table caso seu objetivo seja lidar com o leiaute de sua página.

Há muitos anos, muitos desenvolvedores fizeram uso de tabelas para definir o leiaute de suas páginas. Isso costumava dar origem a código difícil de ler e manter, além de comprometer a semântica do documento. Felizmente, essa fase já foi superada e já sabemos sobre o método tableless.

(WAI-ARIA: Web Accessibility Initiative - Accessible Rich Internet Applications) Até então, temos falado bastante sobre HTML semântico e sobre como o uso de elementos HTML apropriados é importante para a acessibilidade de nossas páginas.

Entretanto, nos dias atuais, aplicações Web que produzem seu conteúdo HTML dinamicamente, ou seja, em tempo de requisição é muito grande. Muitas vezes, esse uso massivo de Javascript torna mais difícil promover a acessibilidade de nossas páginas.

WAI-ARIA é uma especificação que fornece uma coleção de mecanismos que permitem promover ainda mais a acessibilidade de nossas páginas.

A documentação oficial de WAI-ARIA pode ser encontrada a seguir:

<https://www.w3.org/TR/wai-aria-1.1>

Ela faz a seguinte comparação:

Roles, estados e propriedades ARIA são análogos ao CSS para tecnologias assistivas. Ou seja, assim como usamos CSS para explicar ao navegador aspectos visuais da página, usamos ARIA para explicar a estrutura da página às tecnologias assistivas, como leitores de tela.

Nota. O uso de elementos ARIA não têm impacto visual algum. Eles são usados apenas por tecnologias assistivas.

Nota. Não usar ARIA é melhor do que usar ARIA especificado de forma incorreta. Veja esse link:

<https://www.w3.org/WAI/ARIA/apg/practices/read-me-first/>

Essa página cita dois princípios para uso do ARIA. Veja.

Princípio ARIA 1: Uma role é uma promessa.

Quando associamos uma role (papel) a um elemento, estamos fazendo a promessa de que ele, de fato, desempenha aquele papel na página. No código seguinte:

```
<div role="button">Place Order</div>
```

Estamos prometendo que a div desempenha o papel de botão. Escrever um documento HTML que não cumpre essa promessa é semelhante a colocar um botão “Comprar” num site que, quando clicado, deixa o pedido de lado e esvazia o carrinho.

Princípio ARIA 2: ARIA pode ocultar e aprimorar, o que pode ser poderoso e perigoso.

Veja esse exemplo:

```
<a role="menuitem">Assistive tech users perceive this element as an item in a menu, not a link.</a>  
  <a aria-label="Assistive tech users can only perceive the contents of this aria-label, not the link text">Link Text</a>
```

Observe como o uso da role sobrepõe o significado do elemento HTML. O uso de aria-label, por outro lado, aprimora o significado, entregando mais informações para dispositivos de acessibilidade.

Neste próximo exemplo:

```
<button aria-pressed="false">Mute</button>
```

Observe como o desenvolvedor pode explicar textualmente que o botão não está pressionado no momento.

(Elementos progress e meter) Um elemento HTML do tipo **progress** pode ser utilizado para exibir uma barra de progresso ao usuário. O elemento **meter** é semelhante. Veja algumas dicas:

- Usar **progress** para representar o percentual de conclusão de uma tarefa
- Usar **meter** para representar uma medida dentro de um intervalo definido.

Para esse exemplo, crie um arquivo chamado **progress_meter.html**. Veja seu conteúdo inicial a seguir. Começamos com um **progress**. Seus atributos têm o seguinte significado:

- **value**: valor atual
- **max**: valor máximo

```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
  <meta charset="utf-8">
</head>

<body>
  <!-- value: valor atual -->
  <!-- max: valor máximo -->
  <progress value="20" max="100"></progress>

</body>
</html>
```

A seguir, vamos testar o elemento **meter**. Seus atributos têm o seguinte significado:

- **min**: valor mínimo
- **max**: valor máximo
- **low**: valor considerado "baixo"
- **high**: valor considerado "alto"
- **optimum**: valor considerado ótimo
- **value**: valor atual

No exemplo a seguir, definimos diversos elementos **meter** a fim de verificar a forma como cada um é renderizado. Variamos apenas o valor associado ao atributo **value** de cada um:


```
<!DOCTYPE html>
<html lang="pt-BR">

<head>
  <meta charset="utf-8">
</head>

<body>
  <!-- value: valor atual -->
  <!-- max: valor máximo -->
  <progress value="20" max="100"></progress>

  <meter
    min="0"
    max="10"
    low="3"
    high="7"
    optimum="9"
    value="0"
  >
</meter>

  <meter
    min="0"
    max="10"
    low="3"
    high="7"
    optimum="9"
    value="1"
  >
</meter>

  <meter
    min="0"
    max="10"
    low="3"
    high="7"
    optimum="9"
    value="4"
  >
</meter>
```

```

<meter
  min="0"
  max="10"
  low="3"
  high="7"
  optimum="9"
  value="7"
>
</meter>

<meter
  min="0"
  max="10"
  low="3"
  high="7"
  optimum="9"
  value="10"
>
</meter>
</body>
</html>

```


(Microdados) Considere a figura a seguir. Ela mostra o resultado de uma busca feita no Google. Observe que há alguns conteúdos destacados em vermelho.

Google

winston churchill

[All](#)
[Images](#)
[Videos](#)
[News](#)
[Maps](#)
[More](#)
[Tools](#)

About 68,600,000 results (0.63 seconds)

https://en.wikipedia.org/wiki/Winston_Churchill


Winston Churchill - Wikipedia

Sir Winston Leonard Spencer Churchill, KG, OM, CH, TD, DL, FRS, RA (30 November 1874 – 24 January 1965) was a British statesman, soldier and writer who ...

Party: Conservative Party Born: 30 November 1874, Blenheim Pa...

Date of death: 24 January 1965

[Later life of Winston Churchill](#) · [Early life of Winston Churchill](#) · [Clementine Churchill](#)

https://pt.wikipedia.org/wiki/Winston_Churchill · [Translate this page](#)

Winston Churchill – Wikipédia, a enciclopédia livre

Winston Leonard Spencer-Churchill, KG, OM, CH, TD, DL, FRS, RA (Woodstock, 30 de novembro de 1874 – Londres, 24 de janeiro de 1965) foi um político e ...

Precedido por Neville Chamberlain: Primeiro-ministro do Reino Unido

Precedido por Harry S. Truman: Pessoa do ano de 1953


Precedido por Josef Stalin: Pessoa do ano, 1945

Precedido por Clement Attlee: Primeiro-ministro do Reino Unido

People also ask

- Why is Winston Churchill so famous?
- What was Winston Churchill worth when he died?
- How did Winston Churchill lose power?
- What did Winston Churchill win the Nobel Prize for?

Feedback


<https://winstonchurchill.org>


International Churchill Society: Winston Churchill

In his lifetime, Churchill published more than 40 books in 60 volumes, plus hundreds of

Winston Churchill

Former Prime Minister of the United Kingdom



Sir Winston Leonard Spencer Churchill, KG, OM, CH, TD, DL, FRS, RA was a British statesman, soldier and writer who served as Prime Minister of the United Kingdom from 1940 to 1945, during the Second World War, and again from 1951 to 1955. [Wikipedia](#)

Born: November 30, 1874, Blenheim Palace, United Kingdom

Died: January 24, 1965, Kensington, London, United Kingdom




Period: Impressionism

Height: 1.68 m

Children: Randolph Churchill, Diana Churchill, Marigold Churchill, Mary Soames, Sarah Churchill

Dates knighted: October 19, 1922, January 1, 1946, 1953

Artworks [View 10+ more](#)

-  Tower of the Goldfish
-  Daybreak at Cassis
-  The Harbour at Cassis

Tais conteúdos são os **microdados** de uma página. Como desenvolvedores, podemos especificar as informações que desejamos que os buscadores exibam como microdados de nossas páginas.

Para isso, o primeiro passo é escolher um vocabulário, ou seja, uma coleção de nomes ou símbolos que serão utilizados em nossa página e que já são de conhecimento dos buscadores. Assim, quando quisermos explicar que uma div de nossa página representa um filme, por exemplo, temos o símbolo certo para isso, que já será entendido pelos buscadores.

O vocabulário provavelmente mais utilizado hoje é desenvolvido e mantido por grandes empresas como Google, Microsoft e outras. Veja a sua página oficial:

<https://schema.org/>

A página a seguir mostra um exemplo simples::

<https://schema.org/docs/gs.html>

Para este exemplo, crie um arquivo chamado **microdados.html**. Nosso objetivo será fazer a descrição de um filme. Veja:

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

    <meta charset="utf-8">
    <title>Microdados</title>
  </head>
  <body>

    <div>

      <article>
        <h1 >
          A Star is Born
        </h1>
        <p>
          A musician helps a young singer find fame as age and
          alcoholism send his own career into a downward spiral.
        </p>
      </div>
      
      </div>
      <footer>
        Diretor: Bradley Cooper. January 5, 1975.</span>
      </footer>
    </article>

  </div>

</body>
</html>
```

Em nossa página, o elemento **article** representa o filme. Ele será do tipo **Movie** previsto no schema.org. Veja a sua página:

<https://schema.org/Movie>

Observe como associar o tipo Movie ao elemento article:

```
...
<div>

    <article itemscope itemtype="https://schema.org/Movie">

...

```

A página do tipo Movie diz que ele tem algumas propriedades como **name**, **abstract** e **image**. Já temos elementos HTML com esse conteúdo, basta associar os tipos:

```
<article itemscope itemtype="https://schema.org/Movie">
  <h1 itemprop="name">
    A Star is Born
  </h1>
  <p itemprop="abstract">
    A musician helps a young singer find fame as age and
    alcoholism send his own career into a downward spiral.
  </p>
  <div>
    
  </div>

```

Além disso, o tipo Movie tem também uma propriedade chamada **director**. Ocorre que ele é de um tipo específico do schema.org também: o tipo **Person**. Por isso, vamos especificá-lo da seguinte forma:

```

...
<!DOCTYPE html>
<html lang="pt-BR">
  <head>

    <meta charset="utf-8">
    <title>Microdados</title>
  </head>
  <body>

    <div>

      <article itemscope itemtype="https://schema.org/Movie">
        <h1 itemprop="name">
          A Star is Born
        </h1>
        <p itemprop="abstract">
          A musician helps a young singer find fame as age and
alcoholism send his own career into a downward spiral.
        </p>
        <div>
          
        </div>
        <footer itemprop="director" itemscope
itemtype="https://schema.org/Person">
          Diretor: Bradley Cooper. January 5, 1975.</span>
        </footer>
      </article>

    </div>

  </body>
</html>
...

```

Por sua vez, o tipo Person tem as propriedades **name** e **birthDate**. Já temos essas duas informações no rodapé. Entretanto, somente podemos usar o elemento itemprop do HTML5 em elementos HTML. Por isso, vamos englobar cada informação (o nome separado da data de nascimento) em um elemento span do HTML5 que não causa alteração visual alguma. A ele aplicamos os tipos previstos no schema.org. Observe:

```
...
<article itemscope itemtype="https://schema.org/Movie">
  <h1 itemprop="name">
    A Star is Born
  </h1>
  <p itemprop="abstract">
    A musician helps a young singer find fame as age and
    alcoholism send his own career into a downward spiral.
  </p>
  <div>
    
  </div>
  <footer itemprop="director" itemscope
itemtype="https://schema.org/Person">
    Diretor: <span itemprop="name">Bradley Cooper</span>. <span
itemprop="birthDate">January 5, 1975</span>
  </footer>
</article>
span: agrupamento genérico de conteúdos fraseados
...
```

O site schema.org tem um validador de documentos que pode ser acessado a seguir:

<https://validator.schema.org/>

Clique em **CODE SNIPPET** e cole o seu código HTML inteiro. A seguir, clique em **RUN TEST**:

Test your structured data



FETCH URL

CODE SNIPPET

```

1 <!DOCTYPE html>
2 <html lang="pt-BR">
3   <head>
4
5     <meta charset="utf-8">
6     <title>Microdados</title>
7   </head>
8   <body>
9
10    <div>
11
12      <article itemscope itemtype="https://schema.org/Movie">
13        <h1 itemprop="name">
14          A Star is Born
15        </h1>
16        <p itemprop="abstract">
```

RUN TEST

A esperança é que o resultado inclua as propriedades que especificamos e que mostre que não houve erro algum:

Schema.org Documentation Schemas About

NEW TEST

Movie

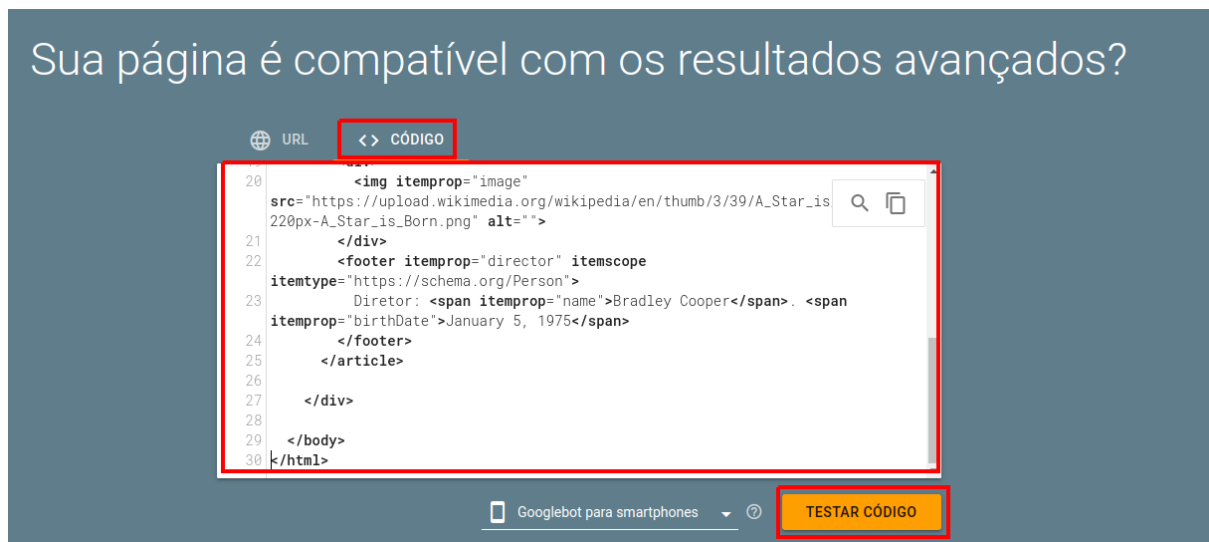
0 ERRORS 0 WARNINGS

@type	Movie
name	A Star is Born
abstract	A musician helps a young singer find fame as age and alcoholism send his own career into a downward spiral.
image	https://upload.wikimedia.org/wikipedia/en/thumb/3/39/A_Star_is_Born.png/220px-A_Star_is_Born.png
director	Person
name	Bradley Cooper
birthDate	1975-01-05

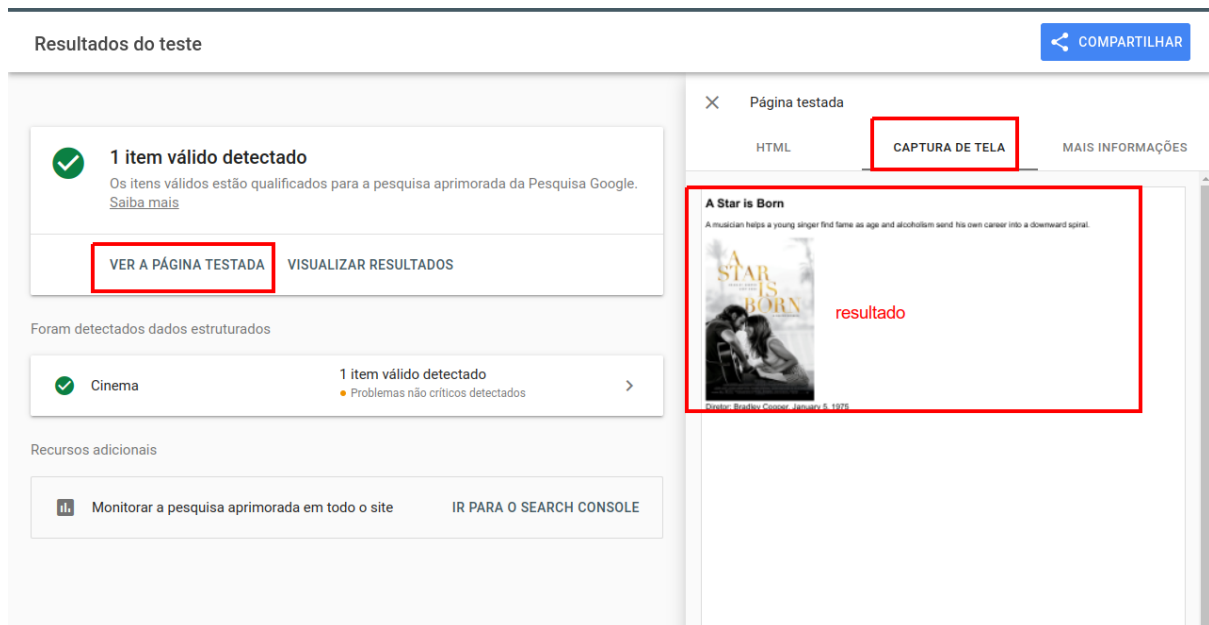
Também pode ser interessante testar o documento HTML e visualizar o conteúdo que seria gerado em função de nossos microdados em páginas de resultados do Google. Isso pode ser feito a seguir:

<https://search.google.com/test/rich-results>

Clique em **CÓDIGO**, cole seu código HTML inteiro e clique em **TESTAR URL**:



A página resultante deverá ser parecida com a seguinte. Clique em **VER A PÁGINA TESTADA**, e, logo a seguir, em **CAPTURE DE TELA**:



Para saber sobre os tipos definidos pelo vocabulário que estamos usando, visite a lista completa:

<https://schema.org/docs/full.html>

: