



# Introdução ao HTML5

FELIPPE ALEX SCHEIDT

HTML



# Introdução ao HTML5

FELIPPE ALEX SCHEIDT

Felippe Alex Scheidt

# Introdução ao HTML5

1ª Edição

Editora Itacaiúnas  
2015

Copyright © 2015, Felipe Alex Scheidt

Capa: Joelson Nascimento

Revisão: Laiza Pâmela Rodrigues Soares Avelino

Editoração eletrônica: Editora Itacaiúnas

S318i
Scheidt, Felipe Alex
Introdução ao HTML5 / Felipe Alex Scheidt 1.ed. – Foz do Iguaçu: Editora Itacaiúnas, 2015.
Ebook: PDF
<b>ISBN</b> 978-85-68154-22-9
1. HTML (HyperText Markup Language) 2. Internet. 3. Websites – desenvolvimento. I. Título.
CDD-004.07
CDU- 004.43

Dados Internacionais de

Catálogo na Publicação (CIP)

O conteúdo desta obra é de responsabilidade do autor, proprietário do Direito Autoral.

## **SOBRE O AUTOR**

Felipe Scheidt é formado em ciências da computação com especialização em desenvolvimento de sistemas web e mestrado em Modelagem e simulação de sistemas. Atua como professor no Instituto Federal do Paraná e leciona na área de desenvolvimento de sistemas para web.

## **SOBRE A OBRA**

Este livro foi elaborado com o objetivo de auxiliar o estudante no seu primeiro contato com a linguagem HTML. Este livro não possui nenhum pré-requisito, e de fato, pode ser estudado por leigos em programação, sendo um ótimo ponto de partida para aqueles que desejam compreender os princípios do funcionamento das tecnologias web. Este livro também foi elaborado tendo como experiência as disciplinas que este autor já ministrou sobre HTML sendo indicado como material de apoio ao professor em disciplinas introdutórias sobre desenvolvimento web.

O autor.

# Sumário

## 1 Introdução

### 1.1 Navegador web

### 1.2 Tecnologias para desenvolvimento web

### 1.3 Editor de texto

### 1.4 Breve introdução à arquitetura web

## 2 Estrutura da HTML

### 2.1 Boas práticas

## 3 Conhecendo as tags

### 3.1 Títulos

### 3.2 Parágrafos

### 3.3 Listas

### 3.4 Imagens

### 3.5 Links

[3.6 A tag <pre>](#)

[3.7 Atributos de tags](#)

#### [4 Adicionando estilo ao HTML](#)

[4.1 A tag style](#)

[4.2 Div](#)

[4.3 A tag span](#)

#### [5 Tabelas](#)

#### [6 Formulários](#)

[6.1 Input](#)

[Atributos](#)

[6.2 Select](#)

[6.3 Textarea](#)

[6.4 Form](#)

#### [7 Metadados e acessibilidade](#)

[7.1 A tag <meta>](#)

[7.2 Técnicas de acessibilidade](#)

[7.3 Atributos data-\\*](#)

#### [8 Novidades na HTML5](#)

[8.1 Simplificações](#)

[8.2 Tags obsoletas na HTML5](#)

[8.3 Novas tags de sessão](#)

[8.4 Tag figure](#)

[8.5 Tags de audio e video](#)

[8.6 Tags UI](#)

[8.6 API selector](#)

[8.7 O elemento canvas](#)

[8.8 LocalStorage](#)

[8.9 API de geolocalização](#)

#### [Apêndice I](#)

[Novas tags da HTML5](#)

[Lista de tags de versões anteriores](#)

#### [Referências](#)

# 1 Introdução

O objetivo deste livro é apresentar de modo introdutório e didático os conceitos básicos de desenvolvimento de páginas web. Seu público alvo são alunos e interessados que não possuem conhecimento da linguagem HTML. Para utilizar este livro é necessário apenas conhecimento básico de informática, como uso da internet, e-mail e algum sistema operacional (Linux ou Windows).

HTML ou *HyperText Markup Language* (linguagem de marcação de hipertexto) é uma linguagem utilizada para escrever e organizar páginas web. O termo *hipertexto* nada mais é do que texto/informação que possui marcações e hiperligações (links) para outros documentos. A HTML é de fato a linguagem básica, o idioma padrão, utilizado por todos os sites disponíveis na internet. Ela é uma linguagem interpretada pelo navegador web, também chamado de *browser*, que faz a leitura do código HTML e gera os componentes gráficos correspondentes. É importante ressaltar que a HTML não é considerada uma linguagem de programação, pois não possui os conceitos fundamentais que linguagens de programação oferecem tais como: 1. variáveis; 2. estruturas de decisão; 3. estruturas de repetição e 4. estruturas de dados. Por isso chamamos a HTML de linguagem de marcação, pois o que basicamente é feito quando escrevemos em HTML é marcar o texto com tags ou etiquetas.

A HTML oferece ao desenvolvedor tags que permitem definir a *estrutura* da página web, como por exemplo: (1) criar listas; (2) exibir vídeos; (3) exibir imagens; (4) criar links; (5) criar hipertextos; (6) definir tabelas e seções; (7) formulários e muitas outras possibilidades.

A HTML surgiu em 1993 sendo publicado como um esboço para a internet pelo pesquisador do CERN Tim Berners-Lee. Com o passar das publicações e novas versões da HTML o consórcio internacional

W3C (*World Wide Web Consortium*) tornou-se o órgão responsável pelas suas atualizações. A seguir vemos um breve histórico do lançamento de cada versão da linguagem HTML:

1993 – Primeira publicação do esboço da HTML.

1995 – publicada a versão 2.0 da especificação.

1997 – publicada a versão 3.2.

1998 – publicada a versão 4.0.

1999 - publicada a versão 4.01.

2014 – publicada a versão HTML5.

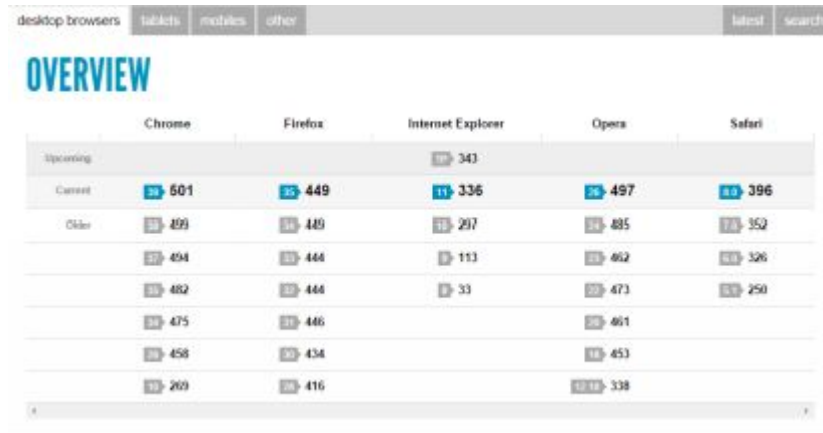
Neste livro abordaremos o conjunto clássico de tags da HTML, presentes desde as versões mais anteriores da especificação HTML. Entretanto, daremos um enfoque especial para as novas funcionalidades da HTML5 em especial as novas tags e atributos. A HTML5 nada mais é do que a última versão da especificação HTML tendo sido finalizada em 2014 e sendo progressivamente implementada pelos navegadores web. Portanto é prudente verificar antes a compatibilidade com os navegadores antes de utilizar funcionalidades do HTML5.

Uma ferramenta útil para verificar o nível de suporte que o seu navegador oferece de HTML5 é o site <http://html5test.com>, no qual o navegador é testado para cada funcionalidade e ao final uma nota é atribuída. Abaixo temos um exemplo do suporte do navegador Chrome na categoria elementos.

Elements		26/30
Embedding custom non-visible data	Yes	✓
New or modified elements		
▶ Section elements	Yes	✓
▶ Grouping content elements	Yes	✓
▼ Text-level semantic elements	Partial	○
download attribute on the a element	Yes	✓
ping attribute on the a element	Yes	✓
mark element	Yes	✓
ruby, rt and rp elements	Yes	✓
time element	No	✗
wbr element	Yes	✓



No mesmo site, podemos conferir o ranking dos navegadores em relação a quantidade de funcionalidades da HTML5 suportadas. Nesta avaliação, vemos que os navegadores Chrome e Opera são os navegadores que mais oferecem suporte à HTML5.



	Chrome	Firefox	Internet Explorer	Opera	Safari
Opening			343		
Canvas	601	449	336	497	396
Audio	499	449	297	485	352
	494	444	113	462	326
	482	444	33	473	250
	475	446		461	
	458	434		453	
	269	416		338	

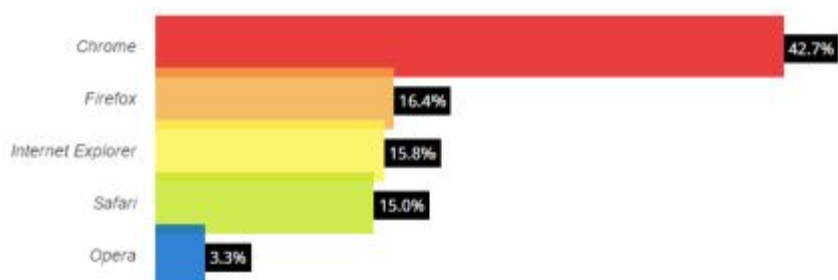
## 1.1 Navegador web

O navegador web ou browser é uma ferramenta indispensável para desenvolvedores *web*. Ele é responsável por interpretar o código HTML e gerar a interface correspondente. Existe uma enorme complexidade nesta tarefa, pois o código HTML inclui também outras tecnologias que trabalham em conjunto, além do fato de que os arquivos estão distribuídos pela internet e precisam ser carregados a cada requisição. O browser precisa assegurar que todos os arquivos sejam posteriormente organizados e interpretados para que o resultado final seja uma página web. Atualmente existem várias opções disponíveis no mercado. Isso por um lado é interessante pois podemos sempre escolher aquele que melhor nos agrada. Por outro lado o resultado de um código HTML nem sempre é exatamente igual de um navegador para outro.

Os principais navegadores disponíveis atualmente são:

1. Google Chrome
2. Microsoft Internet Explorer
3. Mozilla Firefox
4. Opera
5. Safari

Na imagem abaixo extraída do site w3counter podemos ver as estatísticas de uso dos navegadores na internet. Vemos que o navegador Chrome tem se destacado dos demais em termos de utilização em nível mundial, por isso, usaremos aqui este navegador para testar os códigos fontes. Isso não significa que uma vez terminado o seu código HTML você não deva testá-lo nos demais navegadores. De fato, se em algum momento você pensar em publicar seu site na internet, torna-se indispensável testá-lo em todos os tipos de navegadores conhecidos para garantir os mesmos resultados.



Consultado em nov.2014: <http://www.w3counter.com/globalstats.php>

## 1.2 Tecnologias para desenvolvimento web

A HTML não é a única linguagem utilizada para o desenvolvimento de páginas web. Existem diversas outras linguagens coadjuvantes, porém 3 linguagens são fundamentais a qualquer desenvolvedor web:

1. CSS
2. HTML
3. JAVASCRIPT

## 1.3 Editor de texto

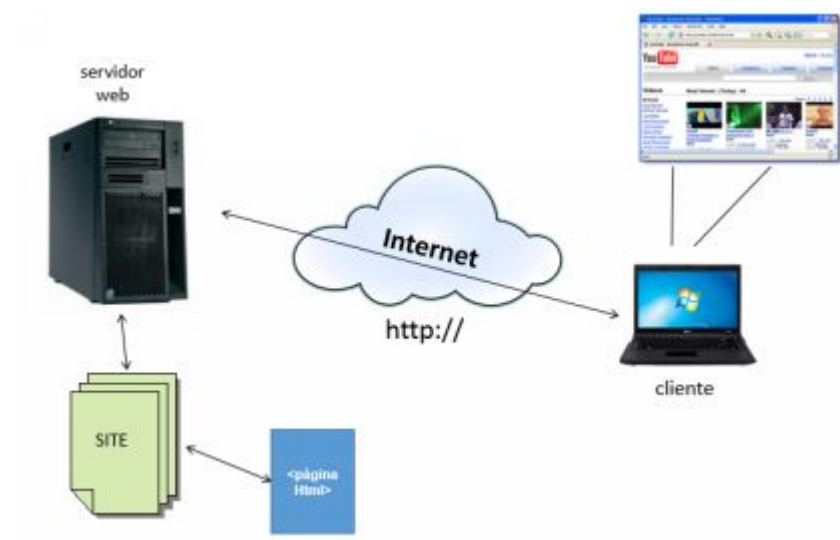
É altamente recomendável que você baixe e instale um editor de texto com suporte à linguagem HTML antes de começar a programar. Evite editores de texto como notepad, writer ou word. Procure editores com suporte HTML. Recomendo começar com ferramentas gratuitas e sem muita sofisticação, pois o objetivo aqui

é aprender a linguagem e não usar a ferramenta. Por isso não recomendo usar logo de início ferramentas como dreamweaver, fireworks e outros similares, pois são ferramentas que trabalham muito a prática do *arrastar-e-soltar*. Nosso objetivo aqui é aprender a lidar diretamente com o código fonte. Assim recomendo as seguintes ferramentas:

1. Notepad++ <http://notepad-plus-plus.org/>
2. Gedit - <http://sourceforge.net/projects/gedit/>
3. HTML-Kit - <http://www.htmlkit.com/download/>

## 1.4 Breve introdução à arquitetura web

A imagem abaixo ilustra a dinâmica de uso de uma aplicação web. Em primeiro lugar, as páginas ou arquivos de um website precisam ser hospedados num servidor, isto é, num espaço online que seja acessível por uma URL. A partir dessa URL máquinas clientes (notebooks, tablets, smartphones, etc) podem acessar o website. O servidor web é a máquina responsável pelo armazenamento do site e por gerenciar conexões de requisição realizadas pelos clientes para determinadas páginas. Uma vez que uma requisição alcança o servidor, o mesmo basicamente consulta seus arquivos, eventualmente acessa uma base de dados, e retorna uma resposta para a máquina cliente. Esse padrão configura-se num esquema requisição-resposta. Esse ping-pong entre solicitação e resposta prossegue até que todos os arquivos necessários para funcionamento do site tenham sido baixados. O protocolo HTTP é utilizado como padrão para troca de dados entre máquina servidor e cliente.



## 2 Estrutura da HTML

Tags são palavras ou termos delimitados por parênteses angulares <>. Por exemplo, a tag <html> define o início da página web e a tag </html> define que a página termina. Todas as tags utilizadas na HTML são palavras reservadas que são conhecidas por todos os navegadores web (a lista completa pode ser vista nos apêndices I e II).

A maioria das tags são encontradas em pares como a tag <html> </html>. É importante conhecer quais tags funcionam em pares pois no momento da abertura de uma nova tag precisamos ficar atento para fechá-la e assim evitar erros.

Vamos começar analisando um exemplo de *hello world* no HTML. Digite o código abaixo no seu editor de texto preferido e salve com o nome hello.html.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Olá Mundo</title>
    <meta charset="utf-8"/>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

A seguir vamos analisar cada uma das linhas do código acima:

### <!DOCTYPE html>

Define que a página segue a especificação da linguagem HTML5, a última versão da HTML. Deve ser sempre a primeira linha do arquivo.

### <html>

É a primeira tag de um arquivo HTML. Ela aparece apenas uma vez e deve ser fechada, na última linha do arquivo.

### **<head>**

Ou cabeçalho, é a sessão da HTML que define informações sobre o código, por exemplo, podemos definir um título para a página, qual a codificação de caracteres, quais são os estilos que serão usados, entre outros.

### **<title>**

Toda página HTML deve possuir um título. Este título aparece na aba do navegador, sendo também exibida no resultado dos mecanismos de busca.

### **<meta>**

Define o tipo de codificação dos caracteres. Como o idioma português possui caracteres especiais como ç e acentuação, que não existem no idioma inglês, usamos a codificação utf-8 que suporta uma maior quantidade de representação de caracteres.

### **<body>**

Essa é uma importante tag que define onde começa o código HTML que tão logo interpretado será exibido no navegador web .

### **<p>**

Essa tag representa um parágrafo de texto.

Como podemos ver a sintaxe (forma de escrever) é bastante simples, entretanto algumas boas práticas são recomendáveis para a organização do código e para que outras pessoas compreendam nosso código.

## **2.1 Boas práticas**

A seguir veremos algumas boas práticas que são fundamentais para que o nosso código fique bem organizado permitindo assim futuras manutenções e melhorias:

### **Toda tag deve ser fechada (há exceções)**

Como vimos no exemplo do *hello world*, sempre que uma tag é aberta temos que ter a atenção de fechá-la. Exemplos: `<p></p>`, `<html></html>`, `<body></body>`, `<title></title>` e `<head></head>`. No entanto, encontramos exceções na HTML, como no exemplo da tag `<meta>`. Observe que a tag meta está sendo aberta e fechada ao mesmo tempo: `<meta charset="utf-8"/>`. O sinal de barra indica o seu fechamento. Mais adiante veremos quais são as tags que não necessitam de fechamento.

### **Respeite a hierarquia entre tags**

Tags possuem a noção de hierarquia, ou seja, existem tags pais e tags filhas, em diversos níveis. Podemos dizer que a tag `html` é o ancestral mais distante, aquela que contém todas as outras tags, portanto precisamos ficar atentos para não cometer um erro desse tipo:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Títulos</title>
  </head>
  <body>
    <h1>Título h1</h1>
    <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed
    do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut
    enim ad minim veniam, quis nostrud exercitation ullamco laboris
    nisi ut </p>
  </html>
</body>
```

Ou seja, fechar uma tag pai antes da tag filha. No exemplo acima, a tag pai é `<html>` e a tag filha é `<body>`. Veja que nesse caso está sendo fechada a tag pai `</html>` antes da tag filha `</body>`. Esse tipo de erro pode gerar bastante dor de cabeça principalmente quando estamos usando programação javascript. Essa regra vale

para todas as tags da HTML que estão dentro de uma estrutura hierárquica.

### **Procure sempre fazer a indentação do código**

A indentação do código facilita muito a leitura do mesmo. Sem indentação fica difícil a leitura do código e acaba atrapalhando você mesmo e a outras pessoas que possam mexer no seu código. Portanto a cada nova linha de código use a tecla TAB para fazer a indentação. Na imagem abaixo, temos um exemplo de indentação de código. Veja que a tag <html> começa sem recuo na margem esquerda, e à medida que novas tags vão sendo criadas dentro da tag <html> colocamos um nível de recuo. Já a tag <title> terá dois níveis de recuo, pois ela é uma tag filha da tag <head> e portanto é "neta" da tag <html>. Podemos pensar nessa analogia para realizar a indentação do código.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Títulos</title>
  </head>
  <body>
    <div>
      <h1>Título h1</h1>
      <p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
      sed do eiusmod tempor incididunt ut labore et dolore magna
      aliqua. Ut enim ad minim veniam, quis nostrud exercitation
      ullamco laboris nisi ut </p>
    </div>
  </body>
</html>
```

Agora observe como o código acima fica sem indentação:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<title>Títulos</title>
</head>
<body>
<div>
<h1>Título h1</h1>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi ut </p>
</div>
</body>
</html>
```



Veja como a leitura fica mais difícil e a possibilidade de erro aumenta bastante, pois a hierarquia entre as tags não está clara ficando fácil deixar uma tag aberta ou cometer outros erros.

## **A HTML é case-insensitive**

Isso significa que não importa para o navegador web se escrevemos o código em letra maiúscula ou letra minúscula. A tag <BODY> é a mesma coisa que a tag <body> ou ainda <Body>. Entretanto é uma boa prática padronizar a escrita do código em letra minúscula e sempre adotar essa forma. Isso facilita a leitura do código fonte para nós e por terceiros.

## **Código em Latim**

Em muitos exemplos que veremos no livro e também em outros encontrados na internet é comum encontrar o seguinte parágrafo:

*Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.*

Fique tranquilo, não precisaremos aprender latim para compreender a HTML. Acontece que em diversas situações nas quais é preciso demonstrar um conceito da HTML precisamos de um texto para "recheiar" uma tag. Imagine o gasto de tempo para realizar essa tarefa se toda vez que fosse necessário um texto, precisássemos sair à procura pela internet... Por isso muitos desenvolvedores web usam o texto "Lorem ipsum" para popular de texto suas interfaces enquanto a programação do lado do servidor ainda não está funcional. Sem mistérios.

## 3 Conhecendo as tags

Neste capítulo, veremos as principais tags da HTML, que são encontradas na grande maioria dos websites. A especificação da HTML define mais de 120 tags (ver apêndices I e II), entretanto uma pequena parcela delas são utilizadas com frequência.

Tags são palavras reservadas ou pré-definidas que são delimitadas por parênteses angulares <>. Por exemplo, já vimos que <html> é uma tag que representa o início de uma página html. Entretanto <ifpr> não é uma tag e não significa nada para o navegador web, pois ifpr não é uma palavra reservada. Portanto você não pode sair criando suas próprias tags, é necessário utilizar alguma tag que esteja definida na especificação da HTML.

Além disso, as tags possuem atributos. Um atributo é uma palavra também reservada na qual podemos associar um valor, que altera alguma característica da tag. Por exemplo:

```
<p align="center">  
  Texto do parágrafo será centralizado  
</p>
```

Nesse caso, o atributo align permite alterar o tipo de alinhamento do parágrafo. Align é uma palavra reservada, assim como center também é um valor pré-definido no qual o browser compreende automaticamente que o texto dentro desse parágrafo deverá ser alinhado.

### 3.1 Títulos

Um título apresenta um assunto com destaque em relação ao texto. Tem como característica uma letra maior e portanto chama a atenção do usuário para determinado tópico ou notícia. Na imagem

abaixo, temos uma notícia cuja manchete é representada como um título:

**Empresa bancada pelo Google lança teste de DNA no Reino Unido**

DA REUTERS

02/12/2014 @ 14h19

Um controverso kit de teste de DNA pessoal da empresa de genética apoiada pelo Google, a 23andme, foi lançado no Reino Unido nesta terça-feira (2).

O produto oferece aos usuários uma chance de ver se correm riscos de certas doenças genéticas e recebeu aprovação para ser vendido, após conseguir a chamada marca "CE" de garantia de qualidade europeia.

No ano passado, a 23andme concordou em parar de vender seu teste de DNA nos EUA, após a Administração de Alimentos e Drogas dos EUA (FDA, na sigla em inglês) dizer que resultados falsos positivos ou negativos poderiam levar pacientes a tomar atitudes "que induzem à morbidez", como cirurgias desnecessárias.

A companhia está agora buscando aprovação regulatória para o mercado norte-americano.

Na HTML, o conjunto de tags que permitem a definição de títulos são: <h1>, <h2>, <h3>, <h4>, <h5> e <h6>. Observe o código a seguir:

```
<body>
  <h1>Título h1</h1>
  <h2>Título h2</h2>
  <h3>Título h3</h3>
  <h4>Título h4</h4>
  <h5>Título h5</h5>
  <h6>Título h6</h6>
</body>
```

Executando esse código no navegador veremos que o resultado será semelhante à imagem a seguir. A tag <h1> representa o título de maior destaque (maior fonte). Geralmente é utilizada para os principais títulos de um site. Títulos secundários são representados com as tags <h2> e <h3> e assim sucessivamente, na medida em que mais níveis de subtítulos são necessários.



## 3.2 Parágrafos

Parágrafos agrupam um bloco de palavras que representam uma ideia que o autor deseja comunicar. Assim como os títulos, os parágrafos são bastantes usados em sites.

A tag `<p>` define um novo parágrafo ou bloco de texto. Veja o exemplo a seguir:

```
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,  
sed do eiusmod tempor incididunt ut labore et dolore magna  
aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis  
aute irure dolor in euendit in voluptate velit esse cillum dolore  
eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non  
proident, sunt in culpa qui officia deserunt mollit anim id est  
laborum.</p>
```

Observe que ao abrir a tag `<p>`, precisamos fechá-la `</p>` posteriormente.

## Empresa bancada pelo Google lança teste de DNA no Reino Unido

DA REUTERS

02/12/2014 @ 14h19

Um controverso kit de teste de DNA pessoal da empresa de genética apoiada pelo Google, a 23andme, foi lançado no Reino Unido nesta terça-feira (2).

O produto oferece aos usuários uma chance de ver se correm riscos de certas doenças genéticas e recebeu aprovação para ser vendido, após conseguir a chamada marca "CE" de garantia de qualidade europeia.

No ano passado, a 23andme concordou em parar de vender seu teste de DNA nos EUA, após a Administração de Alimentos e Drogas dos EUA (FDA, na sigla em inglês) dizer que resultados falsos positivos ou negativos poderiam levar pacientes a tomar atitudes "que induzem à morbidez", como cirurgias desnecessárias.

A companhia está agora buscando aprovação regulatória para o mercado norte-americano.

Parágrafo

### 3.3 Listas

Listas são componentes que permitem enumerar nomes, palavras ou itens. Na imagem a seguir, podemos ver um exemplo de lista de compras com 4 itens sem uma ordenação específica:



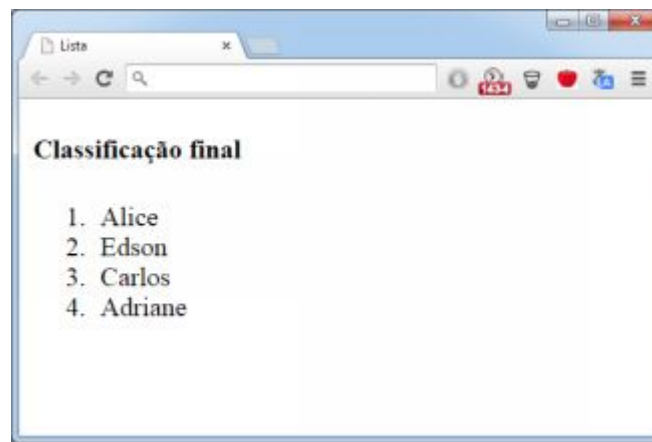
A seguir vemos o código da lista acima:

```
<body>
  <h4>Lista de compras</h4>
  <ul>
    <li>Papel A4</li>
    <li>Caneta</li>
    <li>Lápis</li>
    <li>Caderno 96 folhas</li>
  </ul>
```

```
</body>
```

Observe o uso da tag `<ul>` em conjunto com a tag `<li>`. A tag `<ul>` define o início de uma lista sendo que cada `<li>` define um item da lista (*list item*). A tag `<ul>` significa *unordered list*, ou seja, uma lista sem ordenação.

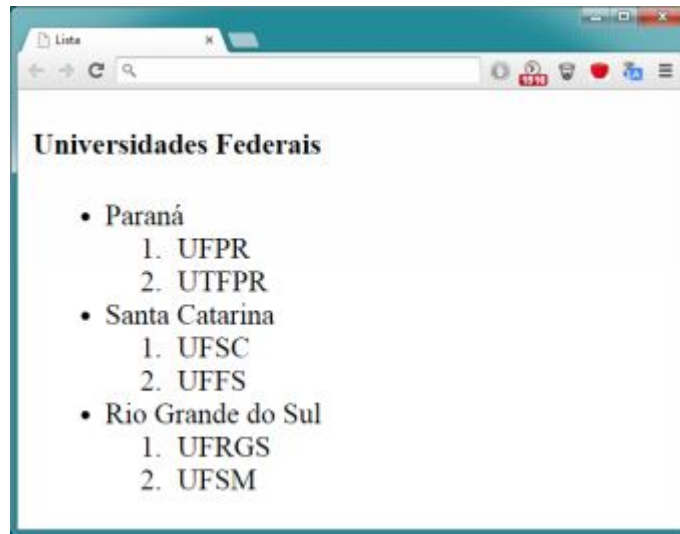
Existem casos em que é necessário haver uma ordenação dos itens de uma lista, conforme imagem abaixo:



```
<h4>Classificação final</h4>
<ol>
  <li>Alice</li>
  <li>Edson</li>
  <li>Carlos</li>
  <li>Adriane</li>
</ol>
```

A única diferença é que aqui usamos a tag `<ol>`, que significa lista ordenada (*ordered list*). O conteúdo da `<ol>` continua sendo composto de `<li>`s.

Outra possibilidade é criar uma combinação entre listas, chamadas listas encadeadas. Considere a seguinte imagem:



Temos um exemplo de lista com 3 estados, não ordenada, contendo uma sub-lista com o nome das universidades federais daquele estado. O código HTML dessa lista é:

```
<h4>Universidades Federais</h4>
<ul>
  <li>Paraná
    <ol>
      <li>UFPR</li>
      <li>UTFPR</li>
    </ol>
  </li>
  <li>Santa Catarina
    <ol>
      <li>UFSC</li>
      <li>UFFS</li>
    </ol>
  </li>
  <li>Rio Grande do Sul
    <ol>
      <li>UFRGS</li>
      <li>UFSM</li>
    </ol>
  </li>
</ul>
```

## 3.4 Imagens

A tag <img> permite anexar uma imagem ao documento HTML. Para isso, basta definir o atributo src (*source*) e apontar para o

endereço onde está localizado o arquivo da imagem:

```

```

Lembrando que o src nesse caso vai apontar para um arquivo, portanto o arquivo logotipo.png deverá estar na mesma pasta que o arquivo html. Caso contrário o src deverá ter o caminho com os diretórios que levem até o arquivo.

Também é possível definir o src como uma URL:

```

```

Os formatos de imagens mais comuns utilizados na construção de páginas HTML são o JPG, PNG e GIF. Em geral o formato JPG é utilizado para representar fotografias, pois necessita de uma grande faixa de representação de cores e boa compactação uma vez que fotografias costumam gerar arquivos grandes. O padrão PNG também pode ser usado para fotografias, entretanto sua taxa de compressão nesse caso não é tão boa quanto ao JPG. O padrão PNG substitui o formato GIF e trabalha com uma compactação muito boa para imagens com pouca variação de cores, como por exemplo 256 cores.

### 3.5 Links

Link ou hiperlink é um recurso da HTML que permite criar um caminho/salto para outra página. Quando o usuário clica em um link o navegador web desvia seu fluxo de navegação para outra página gerando uma nova requisição para o servidor.

Na imagem abaixo, temos um exemplo de uma página com dois links, um para o site institucional do IFPR e outra para o mecanismo de busca google:





O código HTML correspondente é:

```
<h4>Meus atalhos:</h4>
<a href="http://reitoria.ifpr.edu.br">
  Site IFPR
</a>
<a href="http://www.google.com">
  Buscador
</a>
```

A tag <a> representa um link. Entretanto para tornar um link útil precisamos definir o local para onde o mesmo aponta. Usamos o atributo href, passando o endereço de destino. Também definimos a informação que aparecerá para o usuário, ou seja, o texto que se encontra entre <a></a>.

## Definindo o target

Um outro atributo bastante utilizado é o target, que permite definir onde a nova página será carregada. Por padrão quando um link é clicado o conteúdo da nova página é carregado na página atual. Para forçar o navegador abrir a nova página numa nova aba podemos usar o seguinte comando:

```
<a href="http://www.brasil.gov.br" target="_blank">Brasil</a>
```

## Links Internos

Outra possibilidade é criar links para áreas dentro da mesma página. Chamamos estes links de internos. Imagine o caso onde existe uma grande quantidade de texto na página. A navegação pode se tornar difícil e nesse caso muitos sites oferecem um menu que possui links que apontam para sessões dentro da mesma página que, quando clicados, forçam a rolagem da página até o início daquela sessão. Vejamos um exemplo:



```
<h2>Escolha um curso:</h2>
<a href="#biologia">Biologia</a>
<a href="#cinema">Cinema</a>
<a href="#computacao">Ciências da computação</a>

<h3 id="biologia">Biologia</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
</p>
<h3 id="cinema">Cinema</h3>
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,
sed do eiusmod tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud exercitation
ullamco laboris nisi ut aliquip ex ea commodo consequat.
</p>

<h3 id="computacao">Ciências da Computação</h3>
```

```
<p>Lorem ipsum dolor sit amet, consectetur adipisicing elit,  
sed do eiusmod tempor incididunt ut labore et dolore magna  
aliqua. Ut enim ad minim veniam, quis nostrud exercitation  
ullamco laboris nisi ut aliquip ex ea commodo consequat.  
</p>
```

Observando com atenção o código acima, vemos que o "truque" para criação de um link interno consiste em: (1) atribuir um id para cada sessão, que no nosso exemplo é representado pela tag `<h3>` e (2) criar cada link com o sinal de hash `#` junto com o nome da sessão. Logo para criar um link para a sessão computação, basta escrever:

```
<a href="#computacao">Ciências da computação</a>
```

## Links em Imagens

Outro tipo comum de links são aqueles associados a imagens, por exemplos imagens que representam a logo da empresa que quando clicadas redirecionam para uma página com informações da mesma.

```
<a href="about.html" title="saiba mais sobre nós">  
    
</a>
```

É importante que a imagem esteja contida dentro da tag `<a>`. Nesse exemplo, toda a área da imagem será clicável. Observe o uso do atributo `title`. Este atributo funciona como uma dica, quando o mouse passa em cima do elemento, é exibida uma pequena caixa com a mensagem "saiba mais sobre nós".

## 3.6 A tag `<pre>`

O comportamento padrão da HTML é ignorar linhas em branco e espaços. Por exemplo, suponha o seguinte código:

```
<body>  
  <p>Nós somos aquilo que fazemos repetidamente.
```

Excelência, então, não é um modo de agir,  
mas sim um hábito.

ARISTÓTELES

</p>  
</body>

A imagem abaixo mostra o resultado. Vemos que todas as linhas e espaços em brancos foram removidos.



Nesse exemplo, gostaríamos que a frase do parágrafo fosse quebrada em 3 diferentes linhas da mesma forma como escrevemos no código, entretanto, quando o código é interpretado pelo browser, o mesmo automaticamente remove o espaço entre as palavras e as frases. Então como fazer para manter exatamente a mesma formatação que usei no código fonte? Usando a tag <pre>. Reescrevendo o código acima, temos:

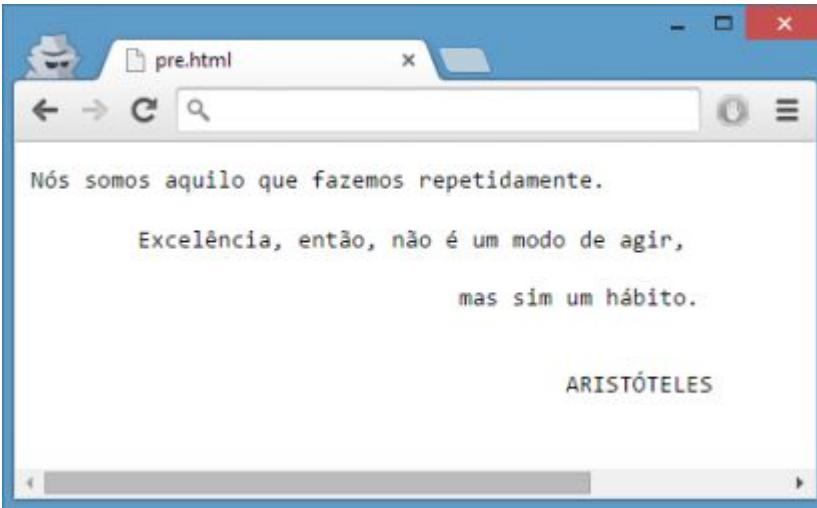
```
<pre>
Nós somos aquilo que fazemos repetidamente.

    Excelência, então, não é um modo de agir,

        mas sim um hábito.

    ARISTÓTELES
</pre>
```

O resultado será muito próximo ao formato que foi escrito no código:



A tag `<pre>` manterá o resultado de modo semelhante ao que foi escrito no código fonte, mantendo todos os espaços e quebras de linha.

### 3.7 Atributos de tags

Conforme já explicamos no início deste capítulo, atributos são recursos valiosos para modificar características das tags. Cada tag pode ter um conjunto de atributos específicos, porém existe um conjunto de atributos globais que se aplicam a todas as tags:

#### **accesskey**

Define uma tecla de atalho para ativar ou colocar o foco na tag.

#### **class**

Define uma classe de estilos do CSS para ser aplicada à tag.

#### **data-\***

Atributo da HTML5 que permite definir um atributo de dados customizável. Isso significa que podemos criar um atributo `data-primeiroNome` que armazenará o primeiro nome do

usuário, por exemplo. O asterisco é definido pelo programador.

## id

Esse atributo permite definir um identificador para a tag. Dessa forma podemos isolar essa tag mais facilmente das outras. A regra mais importante aqui é que o id deve ser único para a tag, isto é, não pode haver duas ou mais tags com o mesmo id.

## style

Esse atributo permite definir uma regra de estilo do CSS diretamente para a tag. Deve ser usado com cautela pois é uma boa prática separar o CSS da HTML.

## title

Esse atributo permite vincular um texto de aviso sobre a tag. Um exemplo típico de uso desse atributo é quando o usuário passa o mouse sobre a tag e um texto informativo aparece como se fosse uma dica.

## ATIVIDADE

1. Faça o código HTML que representa a lista abaixo:



2. Faça o código HTML da lista abaixo, considerando que neste exercício há duas sub-listas:

## Veículos a venda

### 1. **Corsa**

- Motor: 1.0
- Fabricante: GM
- Preço
  - A vista: R\$28.990,00
  - Até 48x: R\$723,00

### 2. **Gol**

- Motor: 1.6
- Fabricante: VW
- Preço
  - A vista: R\$31.877,00
  - Até 48x: R\$911,66

## 4 Adicionando estilo ao HTML

Versões mais antigas da HTML trabalhavam com estilos (cores, fontes, etc) usando atributos das tags. Essa forma de codificação tem sérias desvantagens, pois mistura código de marcação (HTML) com código de aparência (CSS). Para melhor organização de um projeto web, separamos o código de aparência dentro de uma linguagem criada especificamente para isso: Cascading Style Sheet ou CSS. A CSS é uma linguagem totalmente diferente da HTML e possui uma sintaxe própria de funcionamento. A seguir, veremos como criar regras que alteram as cores e tipos de fontes de uma página.

### 4.1 A tag style

O primeiro passo para definir um estilo para a página é criar uma seção de estilos com a tag `<style>` dentro da tag `<head>`:

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Título do site</title>
    <style>

    </style>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

No exemplo acima, para formatar a cor de fundo da página para preto e a cor da fonte para branco, escrevemos as seguintes regras CSS dentro da tag `style`:

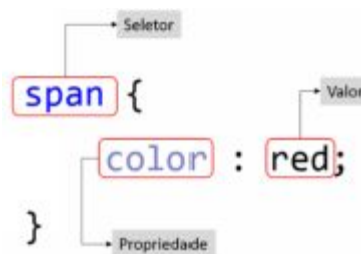


```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"/>
    <title>Título do site</title>
    <style>
      body{
        background-color: black;
        color: white;
      }
    </style>
  </head>
  <body>
    <p>Hello world!</p>
  </body>
</html>
```

No código acima, foi criada uma regra CSS, para a tag body, contendo duas declarações:

1. **background-color: black**  
define que a cor de fundo será preta
2. **color: white**  
define que a cor da fonte será branca

Portanto podemos definir que uma regra CSS é composta de 3 partes: (1) a tag que terá um estilo aplicado, também chamado de seletor; (2) qual a propriedade ou característica que pretendemos modificar e (3) o valor dessa propriedade:



A imagem a seguir mostra a criação de uma regra CSS para a tag span, contendo duas declarações:



The diagram shows the CSS syntax for a `span` element. It starts with `span{` in blue. Inside the curly braces, there are two declarations: `color: #CCC;` and `font-family: verdana;`. Each declaration is enclosed in a red rectangular box. An arrow points from the text "1ª declaração" to the first box, and another arrow points from the text "2ª declaração" to the second box. The code ends with a closing curly brace `}`.

```
span{  
  color: #CCC;  
  font-family: verdana;  
}
```

Fique atento para os seguintes detalhes:

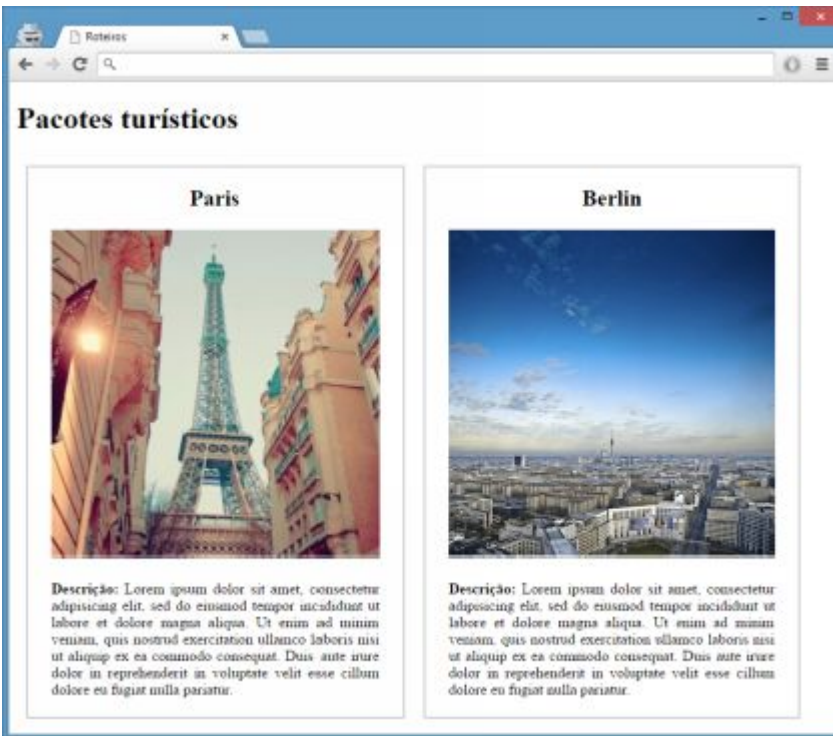
- (1) as declarações devem estar entre chaves;
- (2) a propriedade e o valor são separados por dois-pontos;
- (3) depois do valor, termina-se a declaração com ponto-e-vírgula.

## 4.2 Div

A tag `<div>` fornece um container para agrupar elementos em regiões específicas e assim permitir manejar o layout desse grupo de elementos. Considere a imagem a seguir:



A página trata sobre pacotes turísticos. Cada pacote possui 1 título, 1 imagem e 1 descrição. Individualmente esses elementos não fazem muito sentido. No entanto, nesse caso, agrupar esses 3 elementos dentro de uma div faz sentido, pois a div seria o nosso pacote e além disso poderíamos posicionar melhor cada pacote na tela de modo a aproveitar as regiões em branco. Na imagem a seguir veremos estes elementos agrupados dentro da div:



Cada caixa cinza é uma div. O agrupamento do título, imagem e parágrafo permite que seja aplicado um estilo sobre a div e assim podemos controlar o posicionamento desses elementos de uma forma mais fácil.

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8"/>
  <title>Roteiros</title>
  <style>
    img{
      width: 350px;
    }
    div{
      border: 2px solid #CCC;
      margin: 10px;
      float:left;
      width: 400px;
      text-align:center;
    }
    p{
      width: 350px;
      margin:auto;
      text-align:justify;
    }
  </style>
</head>
<body>
  <div>
    <h3>Paris</h3>
    
    <p>Descrição: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.</p>
  </div>
  <div>
    <h3>Berlin</h3>
    
    <p>Descrição: Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur.</p>
  </div>
</body>
</html>
```

```

padding: 20px 0;
}
</style>
</head>
<body>
<h1>Pacotes turísticos</h1>
<div>
  <h2>Paris</h2>
  
  <p><b>Descrição:</b> Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat.</p>
</div>
<div>
  <h2>Berlin</h2>
  
  <p><b>Descrição:</b> Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod tempor incididunt
ut labore et dolore magna aliqua. Ut enim ad minim veniam,
quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
commodo consequat.</p>
</div>
</body>
</html>

```

## 4.3 A tag span

Essa tag possui uma semântica limpa, sem agregar um estilo embutido como outras tags geralmente possuem. Quando usamos a tag span, é como se não houvesse nenhum efeito visível. Mas é neste momento que podemos tirar vantagens de seu uso. Vejamos o código a seguir:

```

<p>Lorem <span>ipsum dolor</span> sit amet, consectetur
adipisicing elit, sed do eiusmod tempor incididunt ut labore et
dolore magna aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex ea commodo
consequat.</p>

```

Esse código quando interpretado pelo browser não terá nenhuma diferença para um código sem a tag <span>. A principal vantagem dessa tag é a sua ausência de estilo. Isso facilita bastante na hora

de escolher tags para começar um layout do zero. Muitas vezes precisa-se de tags sem formatação para justamente começar a construção do seu próprio estilo e a tag span oferece isso para o web designer. Abaixo temos um exemplo do uso da tag <span> que aplica uma formatação específica em torno de um texto:

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8"/>
  <title></title>
  <style>
    span{
      color: black;
      background-color: yellow;
      border-radius: 5px;
      padding: 5px;
    }
  </style>
</head>
<body>
  <p>Lorem <span>ipsum dolor</span> sit amet, consectetur
adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim
ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat.</p>
</body>
</html>
```

## 5 Tabelas

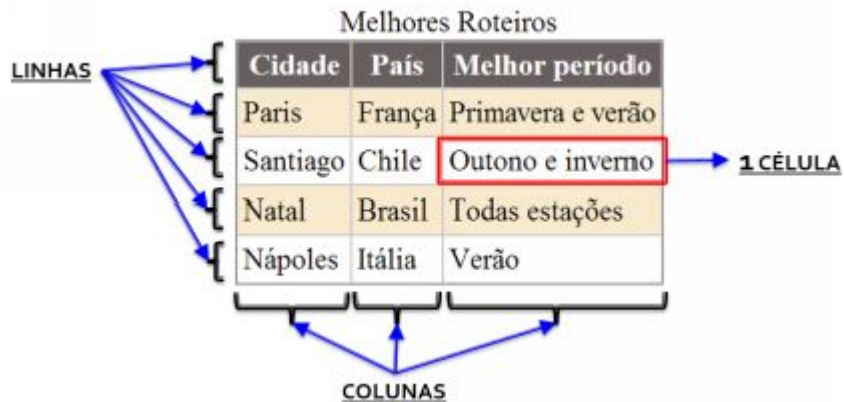
Tabelas são excelentes estruturas para apresentar informações tabulares, ou seja, informações que quando apresentadas em linhas e colunas facilitam bastante a compreensão do texto. Considere o seguinte texto:

*"Em 2010 foram vendidas 128.000 unidades do veículo VW Gol, no mesmo ano foram vendidas 69.900 unidades do veículo GM Corsa e 89.880 unidades do veículo Fiat Palio. "*

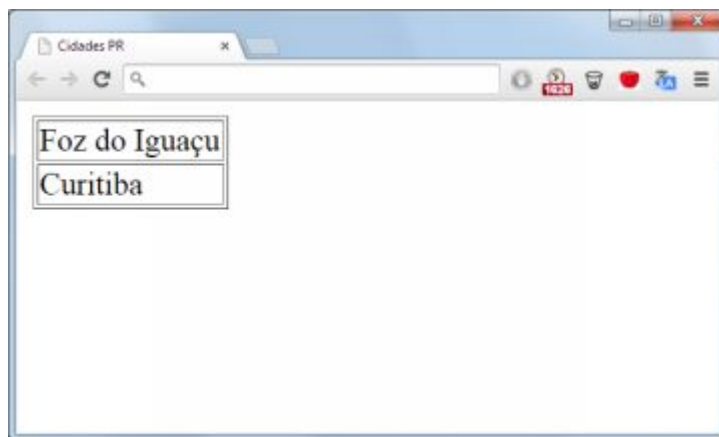
Esse mesmo texto pode ser convertido para a seguinte tabela sem perda de informação:

Vendas de veículos 2010	
Modelo	Vendas
VW Gol	128.000
GM Corsa	69.900
Fiat Palio	89.880

Uma tabela é essencialmente um conjunto de células organizadas em linhas e colunas com o objetivo de apresentar dados de modo mais claro e objetivo. Na imagem abaixo, temos ilustrado as linhas, colunas e células da tabela. Importante notar que na maioria das tabelas temos um cabeçalho, que é a primeira linha da tabela, no qual estão os rótulos ou nomes de cada coluna da tabela. Assim, na tabela abaixo, sabemos que as informações ali apresentadas tratam sobre Cidade, País e Melhor período e por último, no topo da tabela temos uma legenda que informa o assunto da tabela, no caso "Melhores roteiros".



Vejamos agora como criar uma tabela 2x1, ou seja, contendo 2 linhas e 1 coluna, conforme imagem abaixo:



```
<table border="1">
  <tr>
    <td>Foz do Iguaçu</td>
  </tr>
  <tr>
    <td>Curitiba</td>
  </tr>
</table>
```

O código HTML acima apresenta 3 novas tags: <table>, <tr> e <td>.

**<table>**



A tag <table> representa o início de uma nova tabela na HTML. O atributo border="1" torna as bordas da tabela visíveis. Com o uso do CSS não se usa mais esse atributo.

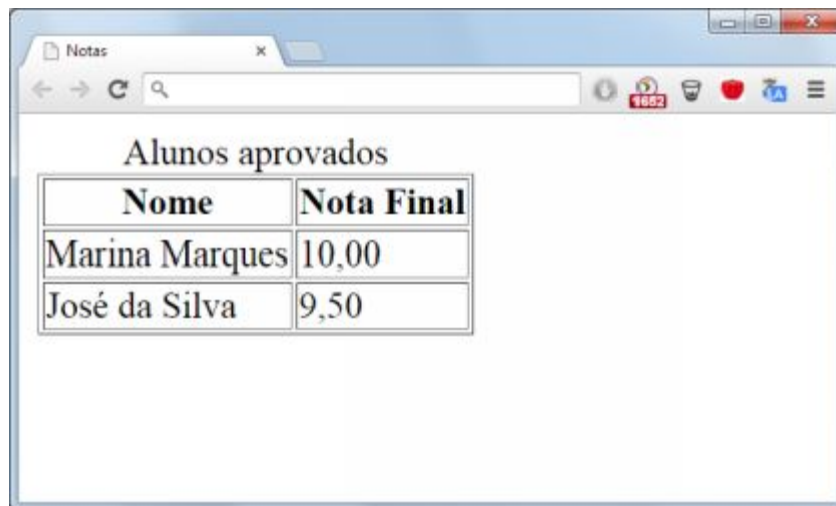
### <tr>

A tag <tr> representa o início de uma nova linha dentro da tabela (*table row*). Sempre que vamos construir uma tabela, começamos contando o número de linhas. Se a tabela possui 4 linhas, teremos 4 tags <tr>.

### <td>

A tag <td> representa uma célula ou *table data*, o que representa também uma coluna.

Agora que já conhecemos como elaborar uma tabela simples, vamos codificar a tabela apresentada na imagem abaixo. Antes de partir para a HTML, analise a imagem buscando identificar o número de linhas, que serão a quantidade de <tr>s e depois o número de colunas, que serão as <td>s. Observe também se existe um título, "Alunos aprovados", na tabela e um cabeçalho (Nome e Nota Final).



Nome	Nota Final
Marina Marques	10,00
José da Silva	9,50

O código da imagem acima pode ser elaborado da seguinte forma:

```
<table border="1">  
  <caption>Alunos aprovados</caption>  
  <tr>  
    <th>Nome</th>
```

```

    <th>Nota Final</th>
  </tr>
  <tr>
    <td>Marina Marques</td>
    <td>10,00</td>
  </tr>
  <tr>
    <td>José da Silva</td>
    <td>9,50</td>
  </tr>
</table>

```

Observe o uso da tag <th>

### <th>

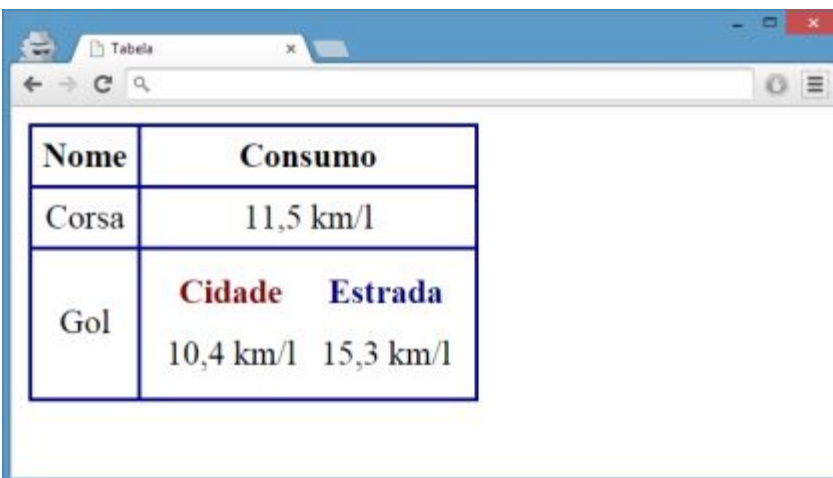
A tag <th> representa os valores do cabeçalho da tabela. É útil para orientar o usuário sobre que tipo de informação determinada coluna possui. Use sempre que possível.

### <caption>

A tag <caption> permite definir um título ou legenda para a tabela. Pode ser posicionada no topo (valor padrão) ou embaixo da tabela.

## Tabelas internas

Da mesma forma que é possível usar listas dentro de listas, podemos usar tabelas dentro de tabelas:



Nome	Consumo				
Corsa	11,5 km/l				
Gol	<table border="1"> <thead> <tr> <th>Cidade</th><th>Estrada</th></tr> </thead> <tbody> <tr> <td>10,4 km/l</td><td>15,3 km/l</td></tr> </tbody> </table>	Cidade	Estrada	10,4 km/l	15,3 km/l
Cidade	Estrada				
10,4 km/l	15,3 km/l				

Veja que a célula Consumo do veículo Gol possui uma estrutura diferente da célula de consumo do veículo Corsa. Essa estrutura é na verdade uma tabela interna, que possui 2 linhas e 2 colunas:

```
<table>
<tr>
  <th>Nome</th>
  <th>Consumo</th>
</tr>
<tr>
  <td>Corsa</td>
  <td>11,5 km/l</td>
</tr>
<tr>
  <td>Gol</td>
  <td>
    <table>
      <tr>
        <th>Cidade</th>
        <th>Estrada</th>
      </tr>
      <tr>
        <td>10,4 km/l</td>
        <td>15,3 km/l</td>
      </tr>
    </table>
  </td>
</tr>
</table>
```

### O atributo colspan

O atributo colspan permite forçar a expansão de uma célula na horizontal. Este recurso se assemelha bastante com a opção mesclar células do software excel. Observe a tabela abaixo:

Desempenho Geral	
Eduardo	884
Jaqueline	856

Esta tabela possui apenas 1 célula na primeira linha, entretanto usando o atributo colspan="2" forçamos que a mesma ocupe o

espaço de duas células:

```
<table>
<tr>
  <th colspan="2">Desempenho Geral</th>
</tr>
<tr>
  <td>Eduardo</td>
  <td>884</td>
</tr>
<tr>
  <td>Jaqueline</td>
  <td>856</td>
</tr>
</table>
```

### O atributo rowspan

O atributo rowspan funciona de modo muito parecido com o colspan. Ele também permite mesclar ou expandir uma célula, entretanto sua expansão é na vertical. Veja a tabela abaixo:

Série A	Times
	Internacional
	Cruzeiro
	Fluminense

A célula "Série A" foi expandida na vertical para ocupar o espaço de 4 células, de acordo com o atributo rowspan=4.

```
<table>
<tr>
  <th rowspan="4">Série A</th>
  <th>Times</th>
</tr>
<tr>
  <td>Internacional</td>
</tr>
<tr>
  <td>Cruzeiro</td>
</tr>
<tr>
  <td>Fluminense</td>
</tr>
```

```
</tr>  
</table>
```

## ATIVIDADE

1. Faça o código HTML da tabela abaixo. Cores utilizadas: yellow; #6E6E6E.

Melhores Roteiros

Cidade	País	Melhor período
Paris	França	Primavera e verão
Santiago	Chile	Outono e inverno
Natal	Brasil	Todas estações

2. Faça o código da tabela abaixo. Para borda use a cor red e o fundo do cabeçalho #CCC.

Consumo na estrada

Nome	Marca	Consumo
Fox	VW	14 km/l
Gol	VW	15 km/l
Golf	VW	11 km/l

3. Faça a tabela abaixo que representa o consumo médio na estrada de diversos veículos:

Consumo estrada		
Imagem	Modelo	Consumo
	Clio	15,8 km/l
	Palio Fire	15,4 km/l
	Novo Uno	15,2 km/l
	Up	14,3 km/l

4. Faça o código da tabela abaixo que representa os imóveis à venda de uma imobiliária. Cores utilizadas: #CCC, #5EFB6E, #CCC e lightblue.

Casas no centro	
	
Imóvel a venda: R\$348.000	Imóvel a venda: R\$680.000
Casas na vila A	
	
Imóvel para aluguel: R\$1200	Imóvel para aluguel: R\$1300

5. Utilizando o conceito de rowspan e colspan, faça o código da tabela a seguir:

Vendas Trimestrais			
2010	144.000	2009	188.000
	139.580		179.580
	254.900		298.500

# 6 Formulários

Formulários são utilizados para coletar dados do usuário, seja através de um texto digitado, de uma opção selecionada ou de um caixa marcada. Existem diferentes formas de interagir com o usuário e coletar suas preferências e informações. A seguir veremos algumas dessas opções.

## 6.1 Input

Input é um tipo de tag multi-uso que genericamente representa uma opção de entrada de dados. O input pode ser configurado de diversas maneiras, de acordo com o seu atributo type, o qual permite representar:

- Data
- Texto
- Horário
- Senha
- Caixa de marcar com múltipla seleção
- Caixa de selecionar apenas uma opção

A especificação da HTML5 fez um grande avanço ao definir novos tipos de entrada de dados, entre os quais, a representação de datas e horário que anteriormente só eram possíveis com o uso do JavaScript.

### Input texto

Esse tipo de input é utilizado para criar um campo de texto, sem restrições, que permite o usuário entrar com qualquer texto. Na imagem a seguir temos um exemplo de uso para criar um campo no



qual o usuário pode digitar seu nome e outro campo que o usuário pode digitar sua data de nascimento:

A screenshot of a web browser window. The browser's address bar shows 'Input[Texto]'. The page content consists of two text input fields. The first field is preceded by the label 'Nome Completo:' and the second field is preceded by the label 'Data de nascimento:'. Both fields are empty and have a light blue border.

O código HTML:

```
Nome Completo:  
<input type="text" name="nome"/>  
Data de nascimento:  
<input type="text" name="nasc"/>
```

Observe o uso do atributo type. Sempre que usamos a tag <input> precisamos definir também qual o seu tipo. No exemplo acima definimos que o type é do tipo text. Isso significa que o campo aceitará qualquer tipo de texto como entrada.

Outro importante atributo que encontramos nas tags <input> é o name. Esse atributo é uma referência a este campo, ou seja, quando sua página for enviada para o servidor e os dados forem processados, como por exemplo, verificar se o usuário possui mais de 18 anos de idade, a forma como o servidor obterá a data de nascimento será através da referência "nasc", pois assim foi nomeada esta input. Lembre-se disso quando estiver programando seus formulários.

## Input checkbox

Esse tipo de input permite criar várias opções do tipo clicável nas quais o usuário pode ou não marcá-las. A imagem a seguir mostra um exemplo de aplicação. Tipicamente usamos checkbox quando uma resposta pode ser composta de várias opções, como é o caso do exemplo abaixo. Nele o usuário pode gostar de mais de um tipo de gênero de filme, inclusive ele poderia gostar de todos os tipos.



O código HTML da imagem anterior:

```
Escolha seu tipo de filmes favoritos?  
<input type="checkbox" name="fav" value="1">Aventura  
<input type="checkbox" name="fav" value="2"/>Comédia  
<input type="checkbox" name="fav" value="3"/>Drama  
<input type="checkbox" name="fav" value="4"/>Ficção  
<input type="checkbox" name="fav" value="5"/>Suspense
```

Observe no código acima que o type é checkbox. Observe também que todos possuem o atributo name igual. O único atributo que varia é o value, pois ele armazena o valor daquela opção. O número 1, 2, 3, 4 e 5 são identificadores para cada opção. Fica a seu critério definir quais valores representarão cada opção, entretanto o mais comum é utilizar aqui o mesmo id que está definido no banco de dados.

## Input radio

Esse tipo de input permite que o usuário selecione apenas 1 opção dentre várias. Existem casos como no exemplo abaixo, no qual é preciso garantir que apenas uma opção esteja selecionada.



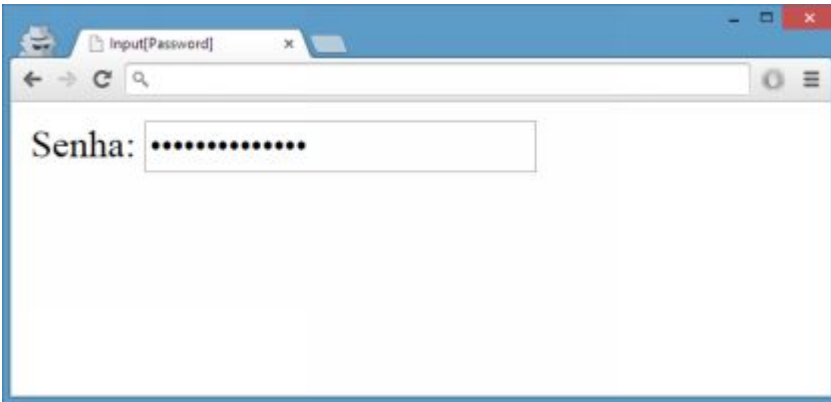
O código HTML:

```
Qual a sua nacionalidade?  
<input type="radio" name="nac" value="BR"/>Brasil  
<input type="radio" name="nac" value="CA"/>Canada  
<input type="radio" name="nac" value="CL"/>Chile  
<input type="radio" name="nac" value="FR"/>França  
<input type="radio" name="nac" value="PY"/>Paraguai
```

Observe que para garantir que apenas uma opção seja selecionada o valor do atributo `name` deve ser igual em todas as opções. Nesse exemplo, ao invés de números, optei por usar a sigla do país no atributo `value` somente para mostrar que tanto letras quanto números são possibilidades de valores.

## Input password

Essa input é usada como campo de entrada de senhas. A senha na HTML é tratada como um campo texto, porém visualmente ela é protegida, ou seja, todo caractere digitado nesse campo é convertido para um caractere padrão (uma bolinha), ficando impossível para algum bisbilhoteiro enxergar a senha digitada, não pelo menos na tela do computador.



O código HTML:

```
Senha: <input type="password" name="senha"/>
```

## Input date

Cria um campo de entrada que aceita apenas datas. Mostra também um calendário para facilitar a seleção da data. O tipo date foi adicionado na HTML5.



Observe que o campo já aparece com uma máscara, isto é, um formato do tipo de dado aceito como entrada. No caso *dd*, significa o dia com dois dígitos, *mm* refere-se a mês também com dois dígitos e *aaaa* representa o ano contendo 4 dígitos.



O código HTML:

```
Data de nascimento:  
<input type="date" name="nascimento"/>
```

Em alguns casos precisamos definir um intervalo de seleção de datas, como uma data de inicial e data final. Usando os atributos *min* e *max* podemos especificar essa validação da seguinte forma:

```
Data de nascimento:  
<input type="date" name="nascimento"  
      min="2012-01-01" max="2014-12-31"/>
```


No exemplo acima, definimos que a data inicial do calendário começa em 01 de janeiro de 2012 e termina em 31 de dezembro de 2014. Observe que a data segue o modelo americano ano-mês-dia.

## Input color

Cria um campo para a seleção de uma cor. Quando clicado, ele mostra uma paleta de cores.

```
Escolha uma cor:  
<input type="color" value="#3B4FC1"/>
```

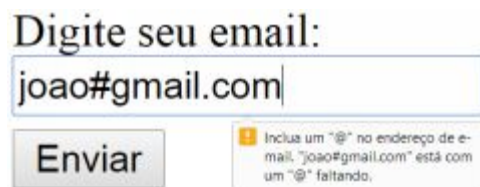
Resultado:

Escolha uma cor: 

## Input email

Cria um campo de texto com regra de validação para emails. O valor inserido pelo usuário deve seguir o formato válido de um email.

```
<form>
  Digite seu email:
  <input type="email" required/>
  <input type="submit"/>
</form>
```



Digite seu email:

joao#gmail.com

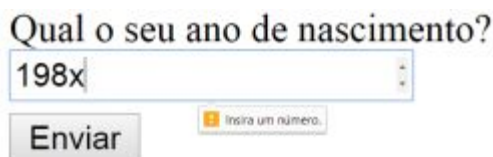
Enviar

Inclus um "@" no endereço de e-mail. "joao#gmail.com" está com um "@" faltando.

## Input number

Cria um campo para entrada de números.

```
<form>
  Qual o seu ano de nascimento?
  <input type="number" required/>
  <input type="submit"/>
</form>
```



Qual o seu ano de nascimento?

198x

Enviar

Insira um número.

O input tipo number aceita os seguintes atributos:

- *min*: define o valor mínimo aceito como entrada.
- *max*: define o valor máximo aceito como entrada.
- *step*: define quanto aumenta o input a cada incremento.

## Input range

Transforma o input numa barra horizontal que limita a entrada de dados apenas para o intervalo representado pela barra de rolagem. Na imagem abaixo, temos uma barra que varia de 0 até 10:

Selecione a quantidade desejada:



```
<p>Selecione a quantidade desejada:</p>  
0  
<input type="range" value="7" min="0" max="10" step="1"/>  
10
```

*min* – define o início do intervalo

*max* – define o fim do intervalo

*step* – define o passo, ou seja, de quantas em quantas unidades é feito o incremento quando o cursor da barra é movido. Por exemplo, se o step fosse igual a 2 a barra poderia ser deslocada para os seguintes valores: 0, 2, 4, 6, 8 e 10.

## Input time

Esse input é especialmente utilizado para representar um horário. Segue o esquema de 24 horas para representação do horário, no formato hh:mm:ss.

Escolha o horário: 19:49

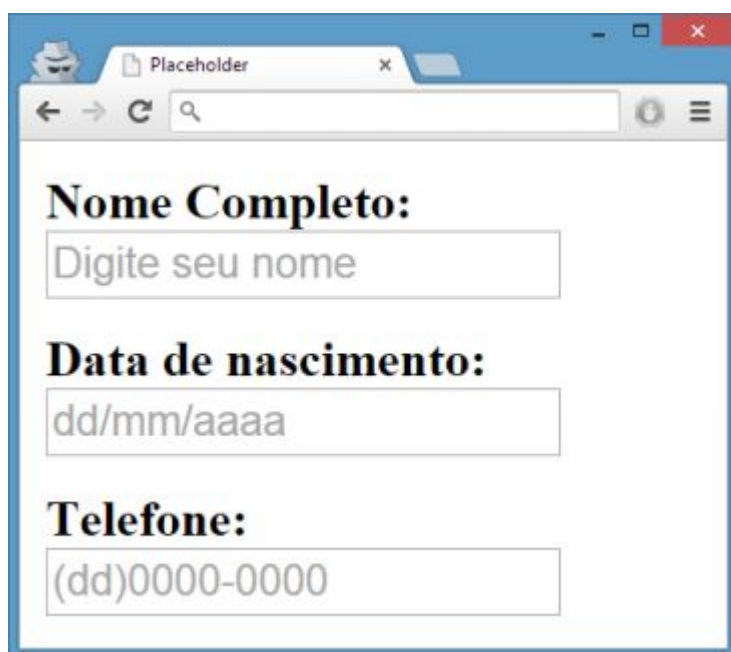
```
<b>Escolha o horário:</b>  
<input type="time"/>
```

# Atributos

A HTML5 também define novos atributos para a tag Input. Destaco as seguintes:

## Placeholder

Permite adicionar uma dica para o usuário sobre que tipo de informação ele deve inserir no campo de entrada, por exemplo:



The screenshot shows a web browser window with the title 'Placeholder'. It contains three text input fields, each with a placeholder text:

- Nome Completo:** Digite seu nome
- Data de nascimento:** dd/mm/aaaa
- Telefone:** (dd)0000-0000

Quando o usuário digita algo, o valor do placeholder desaparece. A dica fica apenas visível enquanto o campo estiver em branco. Além disso, o placeholder não interfere no value do input.

```
<b>Nome Completo:</b>
<input type="text" name="nome" placeholder="Digite seu nome"/>

<b>Data de nascimento:</b>
<input type="text" name="nasc" placeholder="dd/mm/aaaa"/>

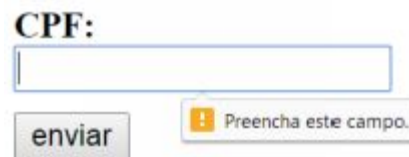
<b>Telefone:</b>
```



```
<input type="text" name="nasc" placeholder="(dd)0000-0000"/>
```

## **Required**

Esse atributo permite fazer uma validação na input. Quando *required* está presente o formulário não será enviado caso o usuário deixe o campo em branco. Essa validação é bem básica e verifica apenas se o campo está ou não em branco.



CPF:

O atributo *required* não requer valor, basta apenas colocar a palavra *required* dentro da tag:

```
<form action="http://www.google.com">  
  <div>  
    <b>CPF:</b>  
    <input type="text" required/>  
    <input type="submit" value="enviar"/>  
  </div>  
</form>
```

## **Autofocus**

Esse atributo força o navegador web a colocar o foco (*cursor*) em um determinado elemento.

```
<b>CPF:</b>  
<input type="text" name="cpf" autofocus/>
```

Assim, toda vez que a página é carregada, o campo cpf ganhará o foco automaticamente. Esse tipo de funcionalidade pode melhorar bastante a usabilidade de uma página que contém um formulário que é frequentemente preenchido por um determinado usuário.

## **Min e Max**

São dois atributos que permitem definir o valor mínimo e valor máximo aceitos como entrada na input. Por exemplo, suponha um cadastro para concurso público no qual o candidato deve possuir uma idade entre 18 e 65 anos:

```
<form action="">
  <b>Qual a sua idade:</b>
  <input type="number" min="18" max="65" required/>

  <input type="submit" value="enviar"/>
</form>
```

## Maxlength

É um atributo que define o tamanho máximo aceito como entrada na input. Por exemplo, suponha que exista um campo *nickname* para o usuário definir seu apelido. Os analistas da empresa definiram que todo *nickname* não pode ultrapassar o tamanho máximo de 20 caracteres. Logo, essa regra pode ser definida na HTML5 com o atributo *maxlength*:

```
<form action="">
  <b>Entre com um nickname:</b>
  <input type="text" maxlength="20" required/>
  <input type="submit" value="enviar"/>
</form>
```

## Autocomplete

O atributo *autocomplete* permite desabilitar a função de autocompletar que os navegadores web oferecem quando o usuário preenche dados. Essa função ajuda em muitos casos nos quais é preciso preencher o mesmo formulário. Entretanto há casos onde é melhor desabilitar a função de autocompletar. Por exemplo, no campo confirme seu email ou inserir número de protocolo é melhor deixar o *autocomplete* desligado para forçar o usuário a preencher o campo novamente.

```
<form action="">
  <b>Digite seu email:</b>
  <input type="email" required />
```

```
<b>Confirme seu email:</b>
<input type="email" required autocomplete="off"/>

<input type="submit" value="enviar"/>
</form>
```

## **Pattern**

O atributo *pattern* (ou padrão) define uma expressão regular que será usada para fazer a validação do input. No exemplo a seguir definimos um *pattern* para validar como entrada apenas uma sequência de caracteres que representa uma placa de veículo. O pattern no caso segue a seguinte expressão regular: [A-Z]{3}[0-9]{4}

[A-Z] – define que aceita qualquer letra maiúscula entre A e Z.

[A-Z]{3} – define que a letra maiúscula deverá aparecer 3 vezes no início da palavra

[0-9] – define que aceita apenas números entre 0 e 9

[0-9]{4} – define que o padrão finaliza com 4 dígitos.

```
<form action="">
  <div>
    <b>Placa do veículo:</b>
    <input type="text" required pattern="[A-Z]{3}[0-9]{4}"/>
    <input type="submit" value="enviar"/>
  </div>
</form>
```

**Placa do veículo:**

AAA111|

enviar

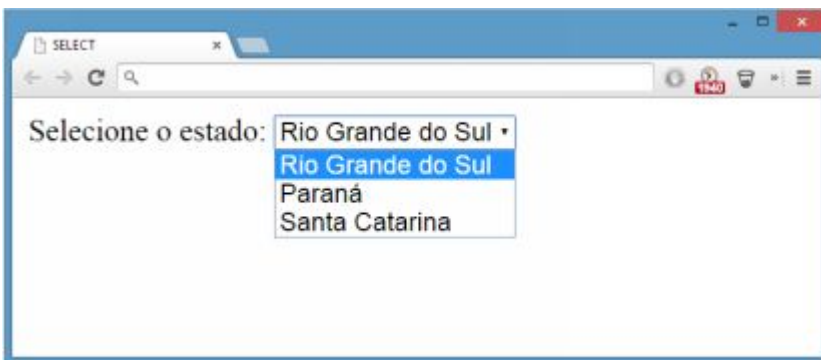
! É preciso que o formato corresponda ao exigido.

## **6.2 Select**

A tag select cria uma lista drop-down com diversas opções para selecionar.



Quando o usuário clica na select, uma lista se expande com todas as opções disponíveis:

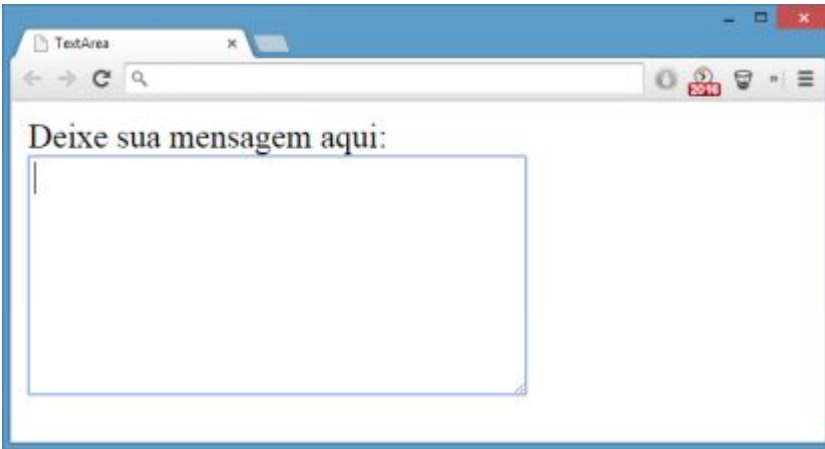


O código HTML:

```
Selecione o estado:  
<select name="estado">  
  <option value="RS">Rio Grande do Sul</option>  
  <option value="PR">Paraná</option>  
  <option value="SC">Santa Catarina</option>  
</select>
```

## 6.3 Textarea

Textarea é uma tag recomendada para a entrada de textos e mensagens longas. Ao contrário da input do tipo text, que exhibe apenas uma linha de texto, a <textarea> pode conter dezenas de linhas.



O código HTML:

```
Deixe sua mensagem aqui:<br>  
<textarea rows="5" cols="30"></textarea>
```

Observe aqui o uso dos atributos rows e cols. Usando o atributo rows podemos definir a quantidade de linhas que a textarea terá, assim como o atributo cols permite definir a quantidade de colunas. Estes valores podem ser alterados de acordo com a sua necessidade.

## 6.4 Form

A tag form não representa um componente visual, ela apenas agrupa um conjunto de campos de entrada de dados (inputs, selects, etc) e define um local no servidor para onde estes dados serão enviados. Abaixo um exemplo de uma típica tela representando um formulário:

A screenshot of a web browser window. The browser's address bar shows a search icon and a red notification badge with the number '2104'. The page title is 'Form'. The main content of the page is a registration form titled 'Novo Usuário'. The form contains three text input fields labeled 'Nome:', 'Email:', and 'Senha:'. Below these fields is a button labeled 'cadastrar'.

O código HTML:

```
<form action="cadastro.php" method="POST">
  <h4>Novo Usuário</h4>
  <label for="nome">Nome:</label>
  <input type="text" name="nome" id="nome"/>

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

  <label for="senha">Senha:</label>
  <input type="password" name="senha" id="senha"/>

  <input type="submit" value="cadastrar"/>
</form>
```

Analisando o código acima vemos que a tag `<form>` é na verdade um *container* que agrupa outras *tags*. A *tag form* possui dois atributos especiais:

### **action**

Esse atributo define o destino final dos dados do formulário, ou seja, para onde os dados serão enviados quando o usuário pressionar o botão "cadastrar". No exemplo acima, o formulário será enviado para a página `cadastro.php` que está hospedada num servidor de aplicações que roda PHP.

## **method**

Esse atributo define a forma como os dados serão enviados. Basicamente temos duas possibilidades de envio: (1) método POST e (2) método GET. No método POST, os dados são enviados no corpo da mensagem HTTP sendo invisíveis para o usuário; e no método GET, os dados são enviados junto com a URL e portanto visíveis ao usuário.

Outra novidade no código anterior é a presença de um input do tipo submit. Em todo formulário é necessária a existência de ao menos uma input desse tipo para disparar o evento de envio do formulário para o servidor. Quando essa input é pressionada, um evento de envio é disparado usando a action como destino e o method como forma de envio dos dados.

# 7 Metadados e acessibilidade

## 7.1 A tag <meta>

A tag meta permite definir os metadados da página web. Metadados são informações sobre dados, ou seja, qual o significado dos dados contidos na página. Por exemplo, numa página web geralmente temos uma quantidade de texto que trata sobre determinado assunto. Como fazer para criar uma descrição geral sobre o conteúdo da página? Uma opção poderia ser resumir o texto e criar uma síntese, mas esse processo seria demasiadamente trabalhoso. A saída é criar um metadado que informa sobre uma descrição geral do site:

```
<head>
  <meta charset="UTF-8">
  <meta name="description" content="Tecnologias sobre
webdesign">
  <meta name="keywords"
content="HTML5,CSS,JQUERY,JavaScript">
  <meta name="author" content="Murilo Silva">
</head>
```

Observe que no exemplo acima, definimos os seguintes metadados:

*charset* – que tipo de codificação (UTF-8) de caracteres é utilizada para representar o texto da página.

*description* – sobre qual assunto a página aborda.

*keywords* – palavras chaves sobre os tópicos e assuntos encontrados dentro da página.

*author* – o nome do autor da página.

É importante observar que a tag meta deve ficar localizada dentro da tag <head>.



## 7.2 Técnicas de acessibilidade

Acessibilidade é um conceito essencial de projeto a ser levado em consideração no momento da construção de um web site. Acessibilidade se preocupa em prover formas diferenciadas de acesso para pessoas com necessidades especiais. Pensando nesse sentido a HTML5 oferece recursos e tags que possibilitam por exemplo que uma pessoa com problemas de visão consiga interagir com a página web e acessar o seu conteúdo.

### **Atributo alt**

O atributo *alt* oferece um texto alternativo para a imagem caso não seja possível carregar o arquivo. O atributo descreve o que se passa na imagem. No exemplo abaixo, caso a imagem não seja carregada, será exibida a mensagem "O avião Airbus A380 faz seu primeiro voo".

```
<figure>
  
</figure>
```

O texto contido no atributo alt também é lido por softwares "leitores de telas" para que pessoas com deficiência visual possam escutar e interagir com a imagem.

### **Atributo title**

O atributo *title* oferece uma dica para o usuário, quando o mesmo posiciona o mouse sobre um elemento.

```
<figure>
  
</figure>
```

O resultado desse código é uma caixa de texto com a mensagem "Este é o novo avião A380 da Airbus" que aparece sobre a imagem quando o mouse é posicionado sobre ela:



O atributo *title* é mais genérico do que o atributo *alt*. Ele pode ser utilizado basicamente em todas as tags para descrever o propósito ou oferecer informações adicionais de um elemento.

### **Atributo Role**

O atributo role permite definir qual o papel ou propósito de uma determinada tag. Apesar de tags como header, footer, article, aside, nav e outras terem uma semântica bem definida, o uso do atributo role visa dar ênfase ao papel específico daquela tag. O uso do atributo role faz parte do conjunto de atributos da ARIA (Accessible rich internet application) que visa atender usuários com algum nível de deficiência, seja auditiva, visual, cognitiva ou locomotiva. Desse modo, quando usamos atributos ARIA estamos tornando a página mais legível para usuários com necessidades especiais que utilizam softwares de leitura de código HTML. Abaixo temos o código de um menu cujo papel é navigation. Softwares que fazem a leitura da página interpretarão a tag <nav> nesse caso como um menu de navegação dando ênfase para o usuário sobre as opções de navegação disponíveis.

```
<nav role="navigation">
  <ul>
    <li>Serviços</li>
    <li>Produtos</li>
    <li>Quem somos</li>
    <li>Contato</li>
  </ul>
</nav>
```

Há casos em que uma mesma tag pode ser utilizada para dois tipos de papéis diferentes e nesse caso podemos explicitar isso com o uso do role. No exemplo abaixo, temos a tag <header> sendo usada para representar o título do site e também para representar o título de um artigo. Com o uso do role, definimos o papel da primeira div como sendo o banner do site e a segunda header como sendo o título do artigo (heading).

```
<header role="banner">  
  Título principal do site  
</header>  
  
<article>  
  <header role="heading">  
    Título ou cabeçalho do artigo  
  </header>  
  <p>Conteúdo principal do artigo</p>  
</article>
```

## 7.3 Atributos data-\*

Atributos data-\* permitem ao desenvolvedor adicionar seus próprios atributos com o propósito de vincular dados a determinados elementos.

```
<ul>  
  <li data-id="19331" data-estado="PR">Roger Silva</li>  
  <li data-id="11414" data-estado="RS">Mariana Toledo</li>  
  <li data-id="6288" data-estado="SP">Lucia Marques</li>  
</ul>
```

No exemplo acima criamos dois atributos data-\*. O primeiro, data-id, poderia ser utilizado para armazenar o id do usuário no sistema. O segundo usamos para representar o estado de origem daquele usuário. Com a especificação data-\* a HTML5 trouxe grande flexibilidade para trabalhar com dados nas tags.

## 8 Novidades na HTML5

Como veremos neste capítulo a HTML5 traz uma série de inovações em relação a sua versão anterior. Esta nova versão da HTML não somente traz novos recursos como também acelera o desenvolvimento de aplicações web disponibilizando ao desenvolvedor novas tags e novos atributos que anteriormente para sua codificação eram necessários o uso de javascript ou CSS. Desse modo a HTML5 deixa o código mais leve e de fácil leitura e entendimento. É importante destacar que a especificação da HTML5 é bastante complexa e abrangente, o que implica que para os próximos anos haverá ainda novidades e consequentemente maior suporte dos navegadores web para todos os itens definidos na especificação HTML5.

Um aviso antes de iniciar o uso da HTML5 é verificar se as tags que você pretende utilizar já estão amplamente implementadas pelos *web browsers*. Recomendo consultar os seguintes sites:

- [www.caniuse.com](http://www.caniuse.com)
- [www.html5test.com](http://www.html5test.com)

Abaixo listo alguns dos pontos que foram trabalhados e enfatizados na especificação da HTML5:

1. Melhor separação entre apresentação e conteúdo: aplicação de estilos feita por CSS apenas, não sendo mais suportado estilos definidos via HTML, como por exemplo, as tags <font>, <big>, <center> e outros.
2. Usar o navegador web para realizar lógica de validação, antes feita apenas por JavaScript.
3. Simplificação do DOCTYPE.

4. A HTML5 resolve boa parte dos problemas que antigamente só eram resolvidos com plugins. Com o uso de funcionalidades nativas da HTML5 + javascript + CSS, consegue-se realizar diversas funcionalidades de modo nativo, como por exemplo, players de vídeo que antigamente exigiam o plugin flash ou windows media player.

## 8.1 Simplificações

A especificação da HTML5 desburocratizou a programação ao simplificar procedimentos bastante repetidos durante o desenvolvimento de uma página. Vejamos alguns:

### DOCTYPE

O DOCTYPE ou tipo de documento é uma instrução para o navegador web que define qual versão da HTML a página foi escrita. Anteriormente o doctype era algo do tipo:

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

passou a ser:

```
<!DOCTYPE html>
```

O doctype acima define que o código da página segue a especificação HTML5. Para versões anteriores da HTML deve-se usar doctype específicos daquela versão.

### TAG STYLE

Na HTML5, o atributo type não é mais necessário para definir o CSS, pois agora é considerado que o type padrão da tag é text/css.

Até a HTML4 era necessário especificar o type do seguinte modo:

```
<style type="text/css">
  body{
    color:red
  }
</style>
```

Na HTML5:

```
<style>
  body{
    color:red
  }
</style>
```

O uso do atributo type passa a ser opcional. Significa que não haverá problema de validação do código se você utilizá-lo.

## CODIFICAÇÃO

Para definir o tipo de codificação de caracteres na página, usava-se anteriormente a tag meta com o seguinte formato:

```
<meta http-equiv="content-type" content="text/html;
charset=UTF-8">
```

Na HTML5, foi simplificado para:

```
<meta charset="UTF-8">
```

## TAG SCRIPT

A tag script também foi simplificada. Até a HTML4 era necessário especificar o type:

```
<script type="text/javascript">
  alert("teste");
</script>
```

Na HTML5, o type passa a ser opcional:

```
<script>
```

```
alert("teste");  
</script>
```

## 8.2 Tags obsoletas na HTML5

A especificação HTML5 tornou obsoleta diversas tags e atributos. Muitas dessas tags foram substituídas por funcionalidades do CSS e novas tags da HTML5. Entretanto tais tags ainda continuam sendo suportadas pelos navegadores web, mas seu uso é altamente desencorajado na HTML5. A seguir, temos a tabela com o nome das tags obsoletas:

acronym	frame	noembed
basefont	frameset	noframes
bgsound	isindex	plaintext
big	listing	rb
blink	marquee	spacer
center	multicol	strike
dir	nextid	tt
font	nobr	xmp

Portanto evite o use de tais elementos, procure alternativas com uso do CSS combinado ao HTML5. Neste site, <https://developers.whatwg.org/obsolete.html>, você pode verificar a lista completa das tags e atributos obsoletos.

## 8.3 Novas tags de sessão

Em versões anteriores da HTML, marcações especiais eram utilizadas para dividir o site em sessões, como por exemplo, o cabeçalho do site, o menu, a parte principal que possui o conteúdo, o rodapé com informações sobre o site, entre outras. Essas sessões costumavam, e ainda costumam ser feitas, usando a tag <div>. Essa tag tem o propósito de representar uma divisão no site, porém não possui uma semântica bem definida, pois não sabemos o significado dessa divisão. Com o objetivo de trazer semântica para o código, a HTML5 propôs a criação de tags específicas para estas sessões que comumente aparecem em todos os sites, que são: título, conteúdo, menu, rodapé, sessões, artigos e cabeçalho. A seguir, veremos cada uma dessas tags.

## **<header>**

Essa tag representa o cabeçalho do site onde tipicamente ficam localizados o título principal do site e algum eventual subtítulo.

## **<section>**

Define uma sessão dentro da página.

## **<article>**

Representa o conteúdo de um artigo, notícia ou post.

## **<footer>**

Representa o rodapé e todo o conteúdo que se encontra nessa sessão.

## **<hgroup>**

Representa uma sessão que contém um grupo de títulos (h1, h2, h3, h4, h5 e h6)

## **<nav>**

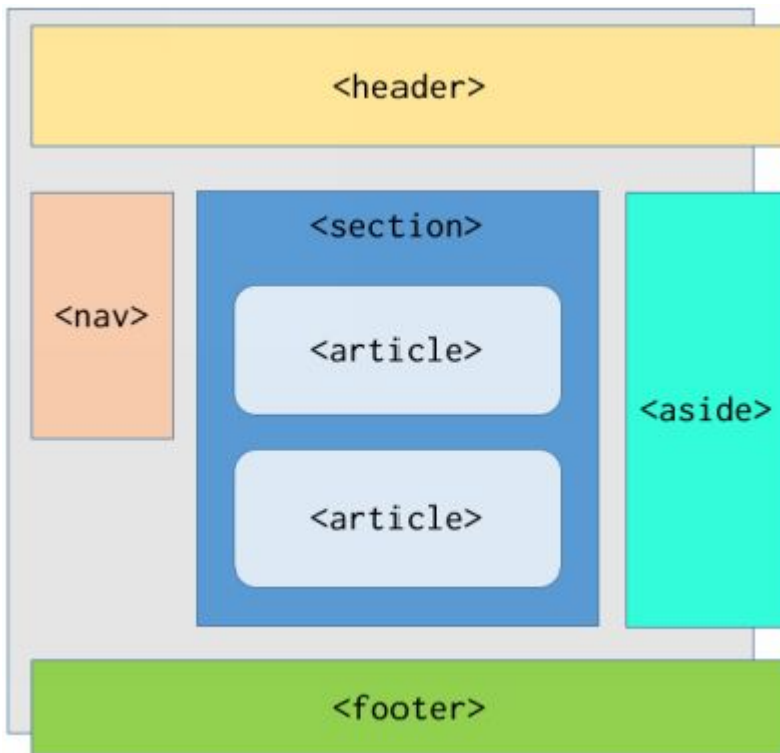
Representa um conjunto de links que permitem fazer a navegação pelo site.

## **<aside>**

Representa um conteúdo relacionado ao assunto da página.

A imagem abaixo mostra um esquema gráfico de aplicação das tags semânticas para estruturação de um site.





## 8.4 Tag figure

A tag `figure` fornece uma semântica para marcar uma foto e legenda no documento. Até versões anteriores do HTML, não havia uma tag com semântica que encapsulasse uma imagem com uma legenda, e para isso tínhamos que usar a tag `div`. Com o uso combinado da tag `<figure>`, `<figcaption>` e `<img>` resolvemos esse problema. Vejamos um exemplo:

```
<h2>Paris</h2>
<figure>
  
  <figcaption>
    Visão da torre Eiffel
  </figcaption>
</figure>
```

Basicamente o que o código acima faz é encapsular uma unidade de conhecimento numa semântica conhecida, no caso uma `figure`

## Paris



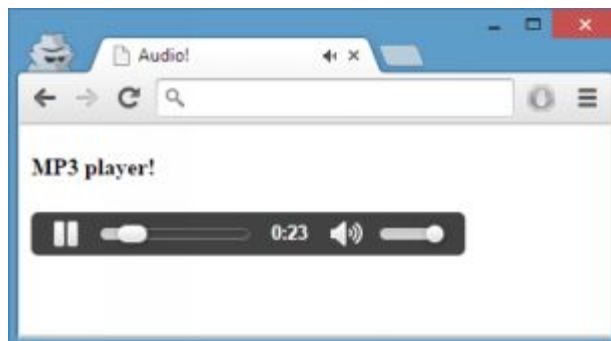
Visão da torre Eiffel

A tag `<figcaption>` define uma legenda para a imagem. Observe que o navegador automaticamente inclui uma formatação CSS para as tags `<figure>` e `<figcaption>`. Essa formatação pode ser alterada criando uma regra CSS.

## 8.5 Tags de audio e video

### AUDIO

Em versões prévias à HTML5, para reproduzir áudio no navegador era necessário ter instalado um plugin. Agora com a tag `<audio>` é possível reproduzir áudio sem a necessidade de plugins. Basta especificar o nome da mídia a ser reproduzida que o próprio navegador, de maneira nativa, cria o componente de controle para isso. O componente oferece funcionalidades básicas como tocar, pausar, avançar, retroceder e controle de volume.



Os atributos mais importantes a serem definidos são:

- *src*: que define o local onde o arquivo de áudio está localizado e o atributo controls

- *controls*: cria a interface que permite controlar o player

```
<h4>MP3 player!</h4>
<audio src="musica.mp3" controls>
  Seu navegador não possui suporte a tag audio
</audio>
```

## **VIDEO**

A tag video permite reproduzir vídeos de maneira nativa sem a necessidade de plugins.



```
<h4>Player de video!</h4>
<video controls src="intro.mp4" width="480" autoplay>
</video>
```

O atributo *src* define o local onde o arquivo de vídeo se encontra. Controls mostra o painel de controle do video, width configura a

largura e autoplay define que o vídeo será reproduzido automaticamente assim que a página for carregada.

## 8.6 Tags UI

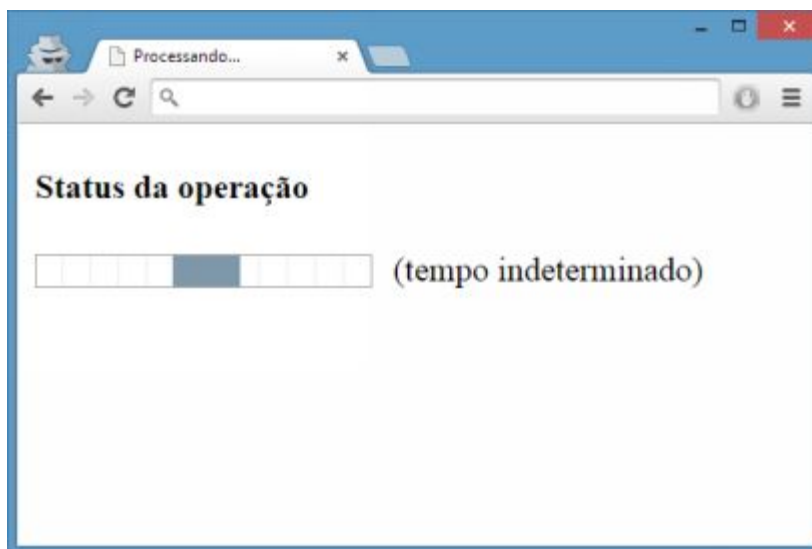
São novas tags da HTML5 que oferecem componentes gráficos de interface com o usuário principalmente para melhorar o feedback da aplicação.

### PROGRESS

A tag `<progress>` cria um componente gráfico que oferece um feedback ao usuário sobre o andamento de uma operação, ou seja, constrói uma barra de progresso para algum processamento que está sendo realizado. Considerando o código abaixo, define-se uma barra de progresso cujo valor máximo é 100, porém sem um tempo determinado:

```
<h4>Status da operação</h4>  
<progress max="100"></progress>(tempo indeterminado)
```

O resultado do código acima é uma barra de progresso que fica oscilando entre as duas pontas:

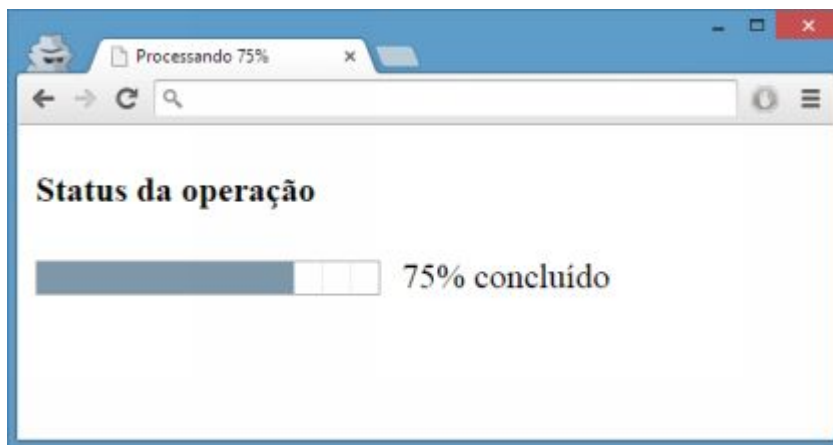


Caso o programador precise definir exatamente em que ponto a barra de progresso está, basta usar o atributo value conforme o

exemplo a seguir:

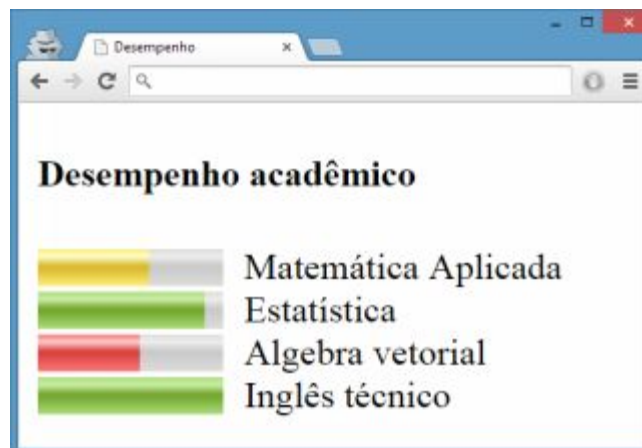
```
<h4>Status da operação</h4>
<progress max="100" value="75"></progress>75%
completado
```

O value pode ser alterado via javascript para criar o efeito de uma barra de progresso que altera-se em tempo real.



## METER

A tag <meter> é utilizada para medir algum dado dentro de um intervalo ou escala de medida. Suponha que precisamos representar as notas de um aluno, que varia de 0 até 100, sendo as notas abaixo de 60 insuficientes, e as notas acima de 90 excelentes:



Para usar uma medida, precisamos definir 6 atributos:

```
<h4>Desempenho acadêmico</h3>

<meter min="0" max="100" low="60" high="90" value="60"
optimum="100">C</meter>
Matemática Aplicada
<br/>

<meter min="0" max="100" low="60" high="90" value="90"
optimum="100">A</meter>
Estatística
<br/>

<meter min="0" max="100" low="60" high="90" value="55"
optimum="100">D</meter>
Álgebra vetorial
<br/>

<meter min="0" max="100" low="60" high="90" value="100"
optimum="100">A</meter>
Inglês técnico
<br/>
```

*min* – é o valor mínimo da escala.

*low* – é o valor que define a partir de onde os valores são considerados baixos. Deve ser maior que *min* e menor que *high*.

*high* – é o valor que define a partir de onde os valores são considerados altos. Deve ser menor que *max* e maior que *low*.

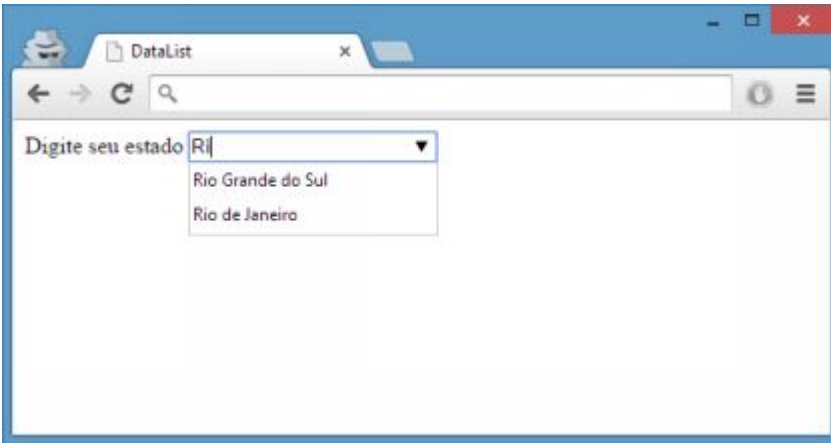
*max* – é o valor máximo da escala.

*value* – é o valor que foi medido e será exibido na régua. É importante observar que o *value* deverá sempre estar dentro do intervalo estabelecido pelos atributos *min* e *max*.

*optimum* - representa o valor ótimo dentro da escala de medida. No nosso exemplo, o valor ótimo é atingir nota 100.

## DATALIST

É uma tag que oferece uma espécie de combo box no qual o usuário digita uma informação numa input e o datalist oferece sugestões de dados, semelhante a uma função de autocompletar.

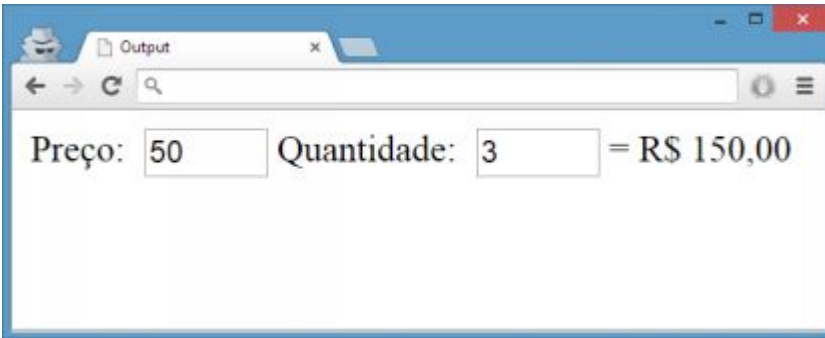


No código abaixo temos uma input que possui um atributo especial "list". Esse atributo vincula a input ao datalist. A vinculação ocorre pelo id do ***datalist*** que deve ser o mesmo nome referenciado no atributo list. O datalist deve ser alimentado com dados na forma de options.

```
<label for="est">Digite seu estado</label>
<input id="est" list="estados"/>
<datalist id="estados">
  <option value="Paraná">
  <option value="Santa Catarina">
  <option value="Rio Grande do Sul">
  <option value="São Paulo">
  <option value="Rio de Janeiro">
</datalist>
```

## **OUTPUT**

A tag output foi criada exclusivamente para apresentar o resultado de alguma operação. Na figura abaixo, temos um exemplo típico no qual precisamos apresentar o total de uma compra para o usuário.



Nesse exemplo, a conta poderia ser realizada por javascript e o total exibido na tag output. Uma das práticas bastante utilizadas consiste em usar um input desabilitado para exibir o total. Entretanto, semanticamente, não é o mais adequado. Por isso para esse tipo de operação usamos a tag output.

```
Preço: <input type="text" value="50">  
Quantidade: <input type="text" value="3">  
=  
<output name="total">R$ 150,00</output>
```

## TIME

A tag time especifica um horário, legível tanto para a máquina quanto para o usuário. O valor definido no atributo datetime comunica ao browser a hora enquanto que o horário definido entre abertura e fechamento da tag será legível para o usuário.

```
As aulas iniciam às <time datetime="19:00">19:00</time> e  
terminam às <time datetime="22:30">22:30</time>
```

## 8.6 API selector

Esta API foi uma grande contribuição para descomplicar a forma como acessamos as tags pelo javascript. Só para ter uma ideia, para selecionar um elemento, poderíamos fazê-lo de 4 maneiras diferentes no javascript: (1) getElementById, (2) getElementsByTagName, (3) getElementsByClassName e (4) getElementsByName.



A fim de evitar confusão, principalmente para quem está começando, a HTML5 tomou emprestada a sintaxe de seletores do CSS para propor um método apenas: `querySelector`.

A sintaxe é bem simples e basta lembrar como funciona no CSS para usar o `querySelector`:

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8"/>
  <title>querySelector</title>
<script>
function teste(){
  // mesmo que getElementById
  var nome = document.querySelector("#nome");
  console.log("Nome: " + nome.value);
  // mesmo que getElementsByClassName
  var cpf = document.querySelector(".cpf");
  console.log("CPF: " + cpf.value);
  // mesmo que getElementsByName
  var rg = document.querySelector("input[name='rg']");
  console.log("RG: " + rg.value);
  // mesmo que getElementsByTagName
  var p = document.querySelector('p');
  console.log("P: " + p.innerHTML);
  // se quiser buscar todos os parágrafos da página:
  var todosParagrafos = document.querySelectorAll('p');
  console.log("Existem " + todosParagrafos.length + "
parágrafos");
}
</script>
</head>
<body onload="teste()">

  Nome:
  <input id="nome" type="text" value="Maria Juliana"/><br>
  CPF:
  <input class="cpf" type="text" value="831.822.983-21"/><br>
  RG:
  <input name="rg" type="text" value="909.134.97-14"/><br>

  <p>Teste de paragrafo 1</p>
  <p>Teste de paragrafo 2</p>

</body>
</html>
```

Observe o uso do método `querySelectorAll` para obter todos os parágrafos da página. O objetivo desse método é retornar todas os elementos que se encaixem com o seletor informado. O método `querySelector` retorna apenas um elemento. Para visualizar o resultado dessa página aperte F12 (Chrome) e depois selecione a aba Console.

### **Obter o elemento cujo id é cpf**

Na HTML4:

```
var cpf = document.getElementById("cpf");
```

Na HTML5:

```
var cpf = document.querySelector("#cpf");
```

### **Obter todos os elementos cuja classe é "obrigatorio"**

Na HTML4:

```
var elementos = document.getElementsByClassName("obrigatorio");
```

Na HTML5:

```
var elementos = document.querySelectorAll('.obrigatorio');
```

### **Obter a input cujo name seja RG**

Na HTML4:

```
var rg = document.getElementsByName("rg")[0];
```

Na HTML5:

```
var cpf = document.querySelector('input[name="rg"]');
```

### **Obter todos os campos textarea**

Na HTML4:

```
var txareas = document.getElementsByTagName("textarea");
```

Na HTML5:

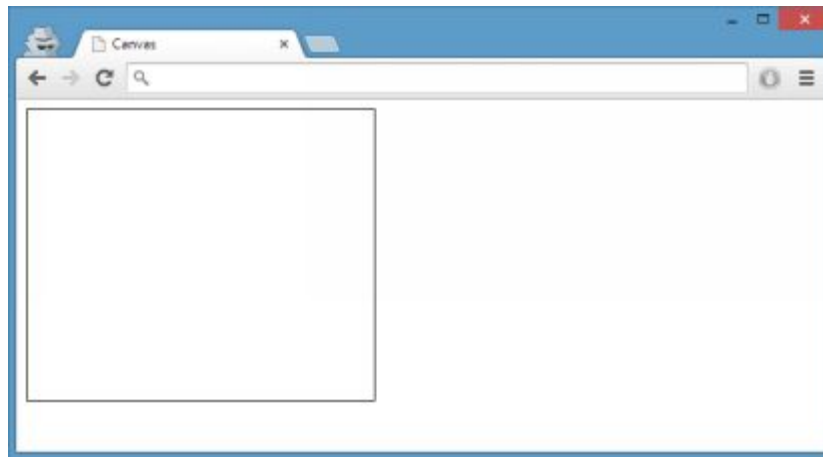
```
var txareas = document.querySelectorAll('textarea');
```

## 8.7 O elemento canvas

A tag <canvas> cria uma área retangular na qual é possível desenhar gráficos, figuras geométricas, textos, imagens e animações utilizando o Javascript. O código HTML para criar um canvas consiste basicamente da tag <canvas> e a definição da largura e altura:

```
<canvas id="area" width="300" height="250"></canvas>
```

O resultado será uma área retangular em branco, com dimensões de 300x250px:



Isso pode ser um pouco frustrante de visualizarmos como resultado, porém não será necessário mais código HTML para obtermos resultados. A partir de agora, usamos Javascript para realizar os desenhos e efeitos especiais.

### Desenhando uma linha

O código javascript abaixo desenha uma linha:

```
1. <body>
2.   <canvas id="canvas" width="600" height="480">
   </canvas>
3.   <script>
4.     var canvas = document.getElementById('canvas');
5.     var context = canvas.getContext('2d');
6.     context.beginPath();
7.     context.moveTo(50, 250);
8.     context.lineTo(400, 30);
9.     context.stroke();
10.   </script>
11. </body>
```

Vamos analisar a sequência de comandos do algoritmo acima:

**Linha 2:** criamos o canvas definindo sua largura e altura.

**Linha 4:** obtemos uma referência ao canvas para poder manipulá-lo.

**Linha 5:** retorna o contexto no qual será realizado o desenho.

**Linha 6:** definimos que a partir desse momento estamos começando o desenho de uma nova figura.

**Linha 7:** a função `moveTo(x,y)` desloca a posição inicial do desenho para as coordenadas 50 no eixo x e 250 no eixo y.

**Linha 8:** esta é a função que constrói a linha. O ponto inicial é (50,250) definido na linha 7 e o ponto final é (400,30).

**Linha 9:** essa é a função que desenha de fato na tela do usuário toda a sequência de comandos anteriores.

## Desenhando múltiplos objetos

No exemplo a seguir, veremos como desenhar duas figuras geométricas, um retângulo e um círculo:

```
<body>
  <canvas id="canvas" width="600" height="480"></canvas>
  <script>
    var canvas = document.getElementById('canvas');
    var context = canvas.getContext('2d');
    // desenha o retangulo
```

```
context.beginPath();
context.rect(320, 120, 150, 100);
context.fillStyle = 'green';
context.fill();
// desenha o círculo
context.globalAlpha = 0.5;
context.beginPath();
context.arc(320, 120, 70, 0, 2 * Math.PI, false);
context.fillStyle = 'blue';
context.fill();
</script>
</body>
```

**context.rect()** – desenha o retângulo. É passado como parâmetro o ponto inicial do retângulo em x (320) e o deslocamento em y (120), a largura (150) e sua altura (100).

**context.fillStyle** – define a cor que será usada para preencher a figura.

**context.fill()** – realiza o desenho da figura.

**context.arc()** – desenha o círculo. Recebe como parâmetro o ponto do centro da esfera no eixo x, y (320, 120), o tamanho da esfera (70) e o ângulo inicial onde começa o desenho e o ângulo final onde termina o desenho.



## 8.8 LocalStorage

Armazenar dados no navegador web sempre foi um desafio com a HTML. Existe a possibilidade de usar cookies ou variáveis de sessão, mas acabam não sendo boas opções quando necessitamos controlar essas informações diretamente do lado do cliente. Visando solucionar essa deficiência, a HTML5 definiu em sua especificação

o objeto LocalStorage. Este objeto pode ser acessado via javascript e permite armazenar dados localmente, mantendo-os mesmo depois do navegador web ter sido encerrado.

O localStorage é basicamente um mecanismo de armazenamento de dados baseado num par chave/valor. Vejamos um exemplo de como armazenar valores na localStorage:

```
<script>
localStorage.setItem("appName","Lista de tarefas");
localStorage.setItem("contato","suporte@todolistnow.com");
</script>
```

Nesse exemplo, usamos o método setItem para armazenar a informação passando como parâmetro o nome da chave e o seu valor. A chave "appName" agora fica vinculada ao valor "Lista de tarefas".

Para recuperar o item usamos o método getItem, passando como parâmetro o nome da chave:

```
<script>
var email = localStorage.getItem("contato");

alert("Email para contato: " + email);
</script>
```

A base do funcionamento dos métodos getItem e setItem gira em torno do par chave/valor. Portanto você precisa se lembrar de suas chaves para poder armazenar e recuperar valores. Outro fator importante é que apenas dados do tipo String podem ser armazenados na localStorage.

## **Lista de tarefas**

Vejamos um exemplo de gerenciador de lista de tarefas bem simples usando o localStorage:

```
<!DOCTYPE html>
<html lang="pt">
<head>
  <meta charset="UTF-8"/>
  <title>TODO List</title>
  <style></style>
```

```

<script>
var lista;
function adicionar(){
    var novaTarefa =
        document.querySelector("#novaTarefa");
    lista = document.querySelector("#lista");
    lista.innerHTML =
        lista.innerHTML + "<li>" + novaTarefa.value + "</li>";
    novaTarefa.value=""; novaTarefa.focus();
}
</script>
</head>
<body>
    <input type="text" id="novaTarefa" placeholder="Digite a
tarefa" autofocus>
    <input type="button" value="cadastrar"
onclick="adicionar()">
    <h4>Lista de tarefas</h4>
    <ul id="lista">
    </ul>
</body>
</html>

```

A atividade agora para você é implementar a funcionalidade que remove um item da lista.

## 8.9 API de geolocalização

A API de geolocalização permite identificar a localização do usuário. A precisão da localização dependerá da capacidade de precisão do dispositivo que o usuário estiver utilizando. Dispositivos com GPS com certeza são os mais precisos, enquanto que aqueles que dependem inteiramente do endereço IP eventualmente apresentarão um erro considerável. Abaixo, temos um exemplo de código que utiliza o objeto `navigator.geolocation` para obter a latitude e longitude através do método `getCurrentPosition()`.

```

<!DOCTYPE html>
<html lang="pt">
<head>
    <meta charset="UTF-8"/>
    <title>Geolocation</title>
<script>
function getLocation(){

```

```

    if (navigator.geolocation){
        navigator.geolocation.getCurrentPosition(
            function(position){
                document.querySelector("#lat").innerHTML =
position.coords.latitude;
                document.querySelector("#long").innerHTML =
position.coords.longitude;
            },function(error){
                document.querySelector("#erro").innerHTML = "ERRO:
" + error.message;
            });
    }
}
</script>
</head>
<body onload="getLocation()">

    <b>Latitude:</b><output id="lat"></output>
    <br>
    <b>Longitude:</b><output id="long"></output>
    <br>
    <output id="erro"></output>

</body>
</html>

```

O código acima precisa ser colocado dentro de um servidor web para funcionar.

A API de geolocalização deve ser utilizada sempre ponderando sobre dois potenciais problemas:

1. A precisão da localização pode ser ruim;
2. O usuário precisa conceder permissão para que a API localize seu dispositivo e portanto não podemos prever se o mesmo aceitará.

Portanto, ao projetar um algoritmo, você precisa construir estratégias alternativas caso algum dos cenários acima aconteça. Você precisa considerar a possibilidade de não obter a latitude e a longitude, prevendo assim uma saída nesse caso.



# Apêndice I

## Novas tags da HTML5

Abaixo são listadas as novas TAGS definidas na especificação da HTML5:

Nome TAG	Descrição
<article>	Pode representar um post de um fórum, um artigo de revista ou jornal.
<aside>	Pode representar uma barra lateral, avisos gerais ou um grupo de links.
<audio>	Permite anexar som a página web.
<bdi>	Bi-directional Isolation – isola uma parte do texto que pode ser formatada em diferentes direções.
<canvas>	Cria uma região na qual é possível fazer desenhos e animações.
<datalist>	Cria uma input com valores pré-definidos
<details>	Define detalhes e informações adicionais sobre algo.
<dialog>	Cria uma caixa de diálogo.
<embed>	Cria um container ou ponto de entrada para um plugin ou aplicação externa.
<figcaption>	Representa uma legenda para uma figure.
<figure>	Cria um conteúdo semântico que representa uma imagem e legenda.
<footer>	Define uma seção que representa o conteúdo do rodapé da página.
<header>	Define uma seção destinada ao conteúdo do cabeçalho da página.
<hgroup>	Agrupa um conjunto de títulos .
<keygen>	Define um campo para geração de chaves.
<main>	Representa o conteúdo principal do documento
<mark>	Cria uma marcação no texto semelhante a uma caneta marca texto, com fundo amarelo e texto preto.

<meter>	Representa um valor escalar dentro de uma unidade de medida.
<nav>	Define uma seção que auxilia na navegação pelo site e suas seções.
<output>	Representa o resultado de um cálculo.
<progress>	Representa uma barra de progresso para feedback ao usuário.
<rp>	Especifica o que exibir caso o browser não suporte anotações ruby.
<rt>	Define uma explicação ou pronúncia de caracteres.
<ruby>	Define uma anotação ruby.
<section>	Define uma seção genérica numa página web.
<source>	Especifica o arquivo de mídia para as tags audio e video.
<summary>	Define um cabeçalho para o elemento details.
<template>	Define um pedaço de código que será utilizado em tempo de execução pelo JavaScript.
<time>	Tag semântica que representa um horário ou horário/data.
<track>	Define uma legenda para um vídeo.
<video>	Tag que permite adicionar um vídeo à página.
<wbr>	Representa uma oportunidade de quebra de linha caso haja pouco espaço.

## Lista de tags de versões anteriores

Abaixo são listadas todas as tags definidas na HTML até a versão 4.01. As tags na cor vermelha e itálico estão obsoletas, ou seja, com o tempo os browsers deixaram de dar suporte as mesmas. A lista completa de tags pode ser consultada aqui: <http://www.w3schools.com/tags/>

Nome TAG	Descrição
<a>	Define um link ou hiperlink.
<abbr>	Representa uma abreviação, podendo conter uma descrição.
<acronym>	Removido da HTML5. Usar a tag <abbr>.

<address>	Usado para fornecer as informações de contato do autor.
<applet>	Removido da HTML5. Usar a tag <object>.
<area>	Define uma região clicável sobre uma imagem, associada a um link.
<b>	Coloca o texto em negrito, usado para destacar palavras-chaves.
<base>	Define uma URL e um target padrão para todos os links na página.
<basefont>	Define a cor e o tamanho da fonte para a página. Não suportado na HTML5.
<bdo>	Bi-directional Override – permite sobrescrever a direção atual do texto.
<bgsound>	<i>Tag obsoleta.</i>
<big>	<i>Tag obsoleta.</i>
<blink>	<i>Tag obsoleta.</i>
<blockquote>	Define um pedaço de texto que é citado de uma outra fonte.
<body>	Define o início do corpo da HTML.
 	Cria uma quebra de linha.
<button>	Cria um botão.
<caption>	Define uma legenda para uma tabela.
<center>	<i>Tag obsoleta.</i>
<cite>	Faz a citação a um título de uma obra.
<code>	Marca um texto como sendo um trecho de código de programação.
<col>	Define um estilo para as colunas de uma tabela.
<colgroup>	Agrupar várias tags <col>.
<dd>	Marca que aquele texto representa uma definição.
<del>	Define um texto que foi deletado.
<dfn>	Representa um termo a ser definido.
<dir>	<i>Tag obsoleta.</i>
<div>	Cria uma divisão ou sessão. Funciona como tag container.
<dl>	Define uma lista de definições.
<dt>	Marca que o texto representa um termo a ser definido, dentro de uma <dl>.
<em>	Coloca em ênfase um texto.
<fieldset>	Agrupar elementos em um formulário.
<font>	Permite alterar o tipo da fonte. Obsoleta.
<form>	Representa um formulário, encapsulando dados enviados para servidor.
<frame>	<i>Tag obsoleta.</i>

<i>&lt;frameset&gt;</i>	<i>Tag obsoleta.</i>
<h1>	Título nível 1.
<h2>	Título nível 2.
<h3>	Título nível 3.
<h4>	Título nível 4.
<h5>	Título nível 5.
<h6>	Título nível 6.
<head>	É uma tag container que agrupa tags <title>, <meta>, <script>, <style> e outras.
<hr>	Cria uma linha horizontal.
<html>	É o elemento raiz da página que representa o início do arquivo HTML.
<i>	Tag que transforma o texto em itálico.
<iframe>	Permite adicionar outro documento HTML à página HTML.
<img>	Permite anexa uma imagem ao HTML.
<input>	Tag que oferece uma entrada de dados para o usuário.
<ins>	Representa um texto que foi inserido recentemente.
<kbd>	Define um texto que representa uma entrada de teclado.
<label>	Cria um texto de rótulo para uma input.
<legend>	Representa um título para o elemento fieldset.
<li>	Cria um item de lista.
<link>	Geralmente usada para linkar uma folha de estilos externa ao documento.
<i>&lt;listing&gt;</i>	<i>Tag obsoleta.</i>
<map>	Define um mapeamento de links sobre uma imagem.
<i>&lt;marquee&gt;</i>	<i>Tag obsoleta.</i>
<meta>	Permite especificar metadados sobre a página.
<i>&lt;nobr&gt;</i>	<i>Tag obsoleta.</i>
<i>&lt;noframes&gt;</i>	<i>Tag obsoleta.</i>
<noscript>	Define um conteúdo alternativo caso o navegador não possua suporte ao JavaScript.
<object>	Cria um objeto dentro do documento HTML que representa um plugin para exibir um arquivo de áudio, vídeo, pdf, flash e outros.
<ol>	Cria uma lista com numeração.
<optgroup>	Faz o agrupamento de grupos de <i>options</i> .
<option>	Define uma opção dentro do <i>select</i> .

<p>	Cria um parágrafo de texto.
<param>	Define um parâmetro para um plugin.
<plaintext>	<i>Tag obsoleta.</i>
<pre>	Preserva a formatação do texto conforme definida no código HTML.
<q>	Define uma citação.
<s>	Risca o texto, o que representa um conceito errado ou irrelevante.
<samp>	Tag usada para destacar um texto que representa uma saída de dados de um computador.
<script>	Delimita uma sessão de código script.
<select>	Cria uma lista drop-down na qual o usuário tem disponível várias opções pré-definidas de escolha.
<small>	Define um texto menor.
<spacer>	<i>Tag obsoleta.</i>
<span>	Tag sem formatação usada para agrupar um conjunto de elementos inline.
<strike>	<i>Tag obsoleta.</i>
<strong>	Coloca ênfase num texto (fonte em negrito).
<style>	Define uma sessão no HTML para inserção de código CSS.
<sub>	Define um texto subscrito.
<sup>	Define um texto sobrescrito.
<table>	Cria uma tabela. Necessita especificar em conjunto as tags <tr> e <td>.
<tbody>	Agrupar os elementos que formam o corpo da tabela.
<td>	Usada dentro de uma tabela, representa uma célula.
<textarea>	Cria uma área para entrada de texto.
<tfoot>	Define o rodapé da tabela.
<th>	Define o cabeçalho de uma coluna.
<thead>	Define o cabeçalho da tabela.
<title>	Atribui um título à página.
<tr>	Define uma linha da tabela ( <i>table row</i> ).
<tt>	<i>Tag obsoleta.</i>
<u>	Sublinha um texto, destacando-o dos demais.
<ul>	Representa uma lista sem numeração.
<var>	Usada para representar uma variável.
<xmp>	<i>Tag obsoleta.</i>



# Referências

1. LUBBERS, Peter; ALBERS, Brian; Salim, Frank. *Programação Profissional em HTML5*. Ed. Alta Books, 2013.
2. ROBSON, Elisabeth; FREEMAN, Eric. *Head First HTML and CSS*. 2ª Edição. Editora O'Reilly, 2012.
3. Mozilla Developer; <https://developer.mozilla.org>
4. W3C Recommendation 28 October 2014; *HTML5: A vocabulary and associated APIs for HTML and XHTML*; <http://www.w3.org/TR/2014/REC-html5-20141028/>
5. W3SCHOOLS; <http://www.w3schools.com/css/>