HTML5 Fundamentos (online)

Cód.: TE 1688/0_EAD

Créditos

HTML5 Fundamentos (online)

Coordenação Geral
Marcia M. Rosa

Coordenação Editorial
Henrique Thomaz Bruscagin

Supervisão de Desenvolvimento Digital
Alexandre Hideki Chicaoka

Produção, Gravação, Edição de Vídeo e Finalização
Xandros Luiz de Oliveira Almeida (Impacta Produtora)
Bruno de Oliveira Santos

Roteirização
Roque Fernando Marcos Souza

Aula ministrada por
Roque Fernando Marcos Souza

Edição e Revisão final
Luiz Fernando Oliveira
Guilherme Yuji Kinoshita

Diagramação
Bruno de Oliveira Santos

Edição nº 1 | 1688/0_EAD
setembro/2014

Este material é uma nova obra derivada da seguinte obra original, produzida por TechnoEdition Editora Ltda., em Jul/2013: HTML 5 Fundamentos
Autoria: Glaucio Daniel Souza Santos

Sobre o instrutor do curso:
Roque Fernando Marcos Souza é consultor em TI, especialista em arquitetura da informação e instrutor de cursos de tecnologia da informação na Impacta Tecnologia desde 1999. Também é palestrante e autor de publicações sobre Web e HTML5.

IMPACTA EDITORA

Sumário

Bem-vindo!

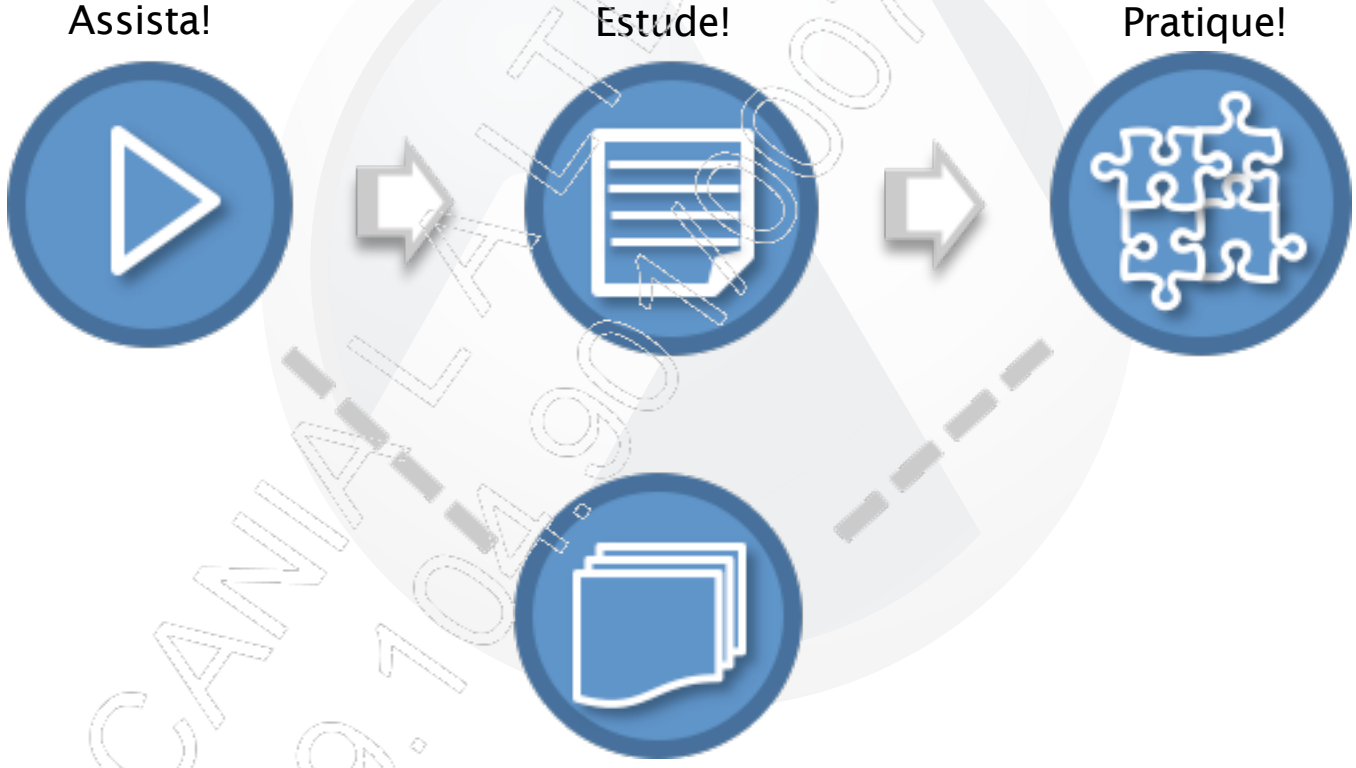
É um prazer tê-lo como aluno do nosso curso online de HTML 5 Fundamentos (online). Este é o curso ideal para você que deseja aprender a criar sites ou web apps (aplicativos web) utilizando a linguagem de marcação HTML, em sua versão 5.

O aprendizado envolve desde os elementos básicos da linguagem até os novos recursos para formulários, passando por recursos para os motores de busca com os novos elementos semânticos, inclusão de imagem, vídeo e áudio.

O curso concede a base necessária para a criação de páginas HTML de forma organizada e contando com o melhor que a tecnologia tem a oferecer.

Para ter um bom aproveitamento deste curso, é imprescindível que você tenha participado do nosso curso de Ambiente Windows, ou possua conhecimentos equivalentes. Também é necessário que tenha conhecimentos de Internet. Bom aprendizado!

Como estudar?



Este curso conta com:

- Videoaulas sobre os assuntos que você precisa saber no curso de HTML5 Fundamentos.
- Parte teórica, com mais exemplos e detalhes para você que quer se aprofundar no assunto da videoaula.
- Exercícios de testes e laboratórios para você pôr à prova o que aprendeu.
- Material de apoio para você testar os exemplos das videoaulas e fazer os laboratórios.

Introdução ao HTML5

1

- ✓ História do HTML;
- ✓ HTML5;
- ✓ Motor de Renderização;
- ✓ Navegadores.

1.1.A história do HTML

HTML (Hypertext Markup Language) é uma linguagem para publicação na Web baseada no conceito de hipertexto, ou seja, elementos que se conectam entre si formando uma rede de informação. Esses elementos podem ser áudio, vídeo, texto etc.

O hipertexto possibilita a comunicação de dados e a organização de conhecimentos e informações. Nesse contexto, a HTML surge como proposta de linguagem universal, entendida por vários meios de acesso, para distribuição global dessa informação.

Tim Berners-Lee desenvolveu a linguagem HTML. A linguagem se tornou popular na década de 1990, quando desenvolvedores e fabricantes de browsers a transformaram em uma linguagem base. Em 1997, com a versão 3.2 desenvolvida pelo W3C, ela se torna uma linguagem padrão.

Desde sua criação, a HTML tem como uma de suas principais características a interoperabilidade, ou seja, pode ser usada em vários dispositivos, plataformas ou outros meios de acesso.

Em 2004, surge o WHATWG (Web Hypertext Application Technology Working Group), grupo formado por especialistas das grandes corporações de TI, que discordava dos caminhos da Web até então e que decide desenvolver uma nova linguagem, uma linguagem mais flexível para Web: HTML5.

Em 2006, houve o reconhecimento do trabalho do grupo. O próprio Tim Berners-Lee se juntou ao WHATWG e, oficialmente, em 2009, foi anunciado o desenvolvimento conjunto da HTML5.

1.2.Markup Languages

As Markup Languages, também conhecidas como MLs, são linguagens que têm o objetivo de fornecer elementos adicionais a um texto, denominados marcações, a fim de definir uma estrutura ou um layout de informações.

A tabela a seguir descreve algumas Markup Languages cujos códigos são compostos não apenas por conteúdo, mas também por tags, que são marcas utilizadas para enfatizar a estrutura de um documento. Observe:

Markup Language	Descrição
HTML (HyperText Markup Language)	Linguagem utilizada com a finalidade de publicar hipertextos na Web. Os hipertextos são documentos que possuem referências internas a outros documentos, isto é, links ou hiperlinks. A HTML é uma linguagem não extensível, que permite descrever documentos que não apresentam complexidade em sua estrutura.
XHTML (Extensible HyperText Markup Language)	Linguagem criada posteriormente à HTML, apresentando uma reformulação desta.
HTML5	Trata-se de uma evolução das versões anteriores da HTML, também com a responsabilidade de criar a marcação de hipertexto, porém, com recursos mais avançados, código mais limpo, ênfase em semântica e interoperabilidade dos sistemas. É basicamente uma combinação de HTML (novos elementos e elementos melhorados) + JavaScript + CSS3.

1.3.HTML5

Esta linguagem é uma evolução dos padrões HTML que já existiam, ela associa as tags de marcação utilizadas em HTML com novos conceitos semânticos e uma coleção de novos recursos nativos que antes eram possíveis somente com plugins.

Para criar vídeos, player de áudio e animações, antes era necessário utilizar plug-ins como Flash, mas com o HTML5 muitos recursos se tornaram nativos e dispensam o uso de plug-ins. A vantagem é que o site ou aplicativo Web (Web App) poderá rodar em vários dispositivos.

Quando um site funciona normalmente em qualquer navegador, chamamos de **cross-browser**. Quando um site funciona normalmente em qualquer dispositivo, como desktop, notebook, tablet, smartphone, TV, carro, óculos, chamamos de **cross-device.**

Quando trabalhamos com a linguagem XHTML (Extensible HyperTex Markup Language), contamos com as tags, as quais são definidas como códigos desta linguagem que têm a função de representar um comando. As tags são colocadas entre os sinais de menor e de maior, da seguinte forma: **<nome_tag>**.

1.4.Visualizando o código fonte

Para conhecermos o código fonte de uma página Web, podemos realizar dois procedimentos distintos: clicar com o botão direito do mouse sobre a página desejada e selecionar a opção **Exibir Código Fonte** (**View Source**) ou, na barra de menu do browser, clicar em **Exibir** (**View**) e, então, em **Código Fonte** (**Source**).

Ao realizar um desses dois procedimentos, o bloco de notas ou outro editor de texto é aberto, trazendo o código XHTML referente ao documento. Quando o arquivo possui extensão **.html**, é possível que o browser exiba imagens, textos, vídeos, filmes, entre outros, assim como é possível que o bloco de notas exiba o código fonte responsável por inserir, diagramar e configurar os elementos.

Devemos ter em mente que, mesmo sendo possível visualizar o código de uma página, não é possível alterar a formatação que ele apresenta. Isso ocorre porque tanto a página quanto o código que visualizamos são cópias dos originais, os quais estão no servidor e são protegidos por senhas e programas contra invasão.

Quando visitamos uma página Web, ocorre o download de arquivos que permanecem armazenados em uma pasta de arquivos temporários da Internet até que sejam excluídos. As imagens, os textos, os arquivos de formatação de estilos, arquivos JavaScript permanecem armazenados nesta pasta, permitindo, assim, que as páginas sejam carregadas no browser usando a cópia já armazenada.

1.5.Interpretação e transformação do código fonte

Ao abrirmos o browser, inserirmos uma URL em seu campo **Endereço** e pressionarmos a tecla ENTER, a resposta obtida é um arquivo de texto simples que pode ser interpretado de várias maneiras, de acordo com a extensão que ele apresenta. Caso a extensão desse arquivo seja **.html**, o browser aciona seu mecanismo de renderização.

A engine é responsável não apenas pela interpretação do código fonte, mas também por sua transformação em elementos gráficos.

O processo de renderização é ato da transformação do código fonte. Este termo é bastante utilizado para falar a respeito de métodos que permitem calcular cores, sombras e texturas, seja em um documento baseado em MLs ou em uma imagem vetorial.

1.6.Navegadores

Existem atualmente centenas de browsers no mercado e qualquer desenvolvedor que tenha conhecimento da estrutura de renderização e interpretação de códigos HTML pode criar o seu navegador.

Podemos mencionar os principais navegadores existentes no mercado. O que deve ficar claro é que todo navegador de Internet possui em seu núcleo um motor de renderização, que é responsável por determinar como o navegador interpreta a árvore de elementos de um código HTML.

1.6.1. Motor de Renderização

Existem alguns motores de renderização que são softwares utilizados no núcleo dos navegadores e clientes de e-mail responsáveis por interpretar a linguagem HTML, folhas de estilo (CSS), imagens e gerar um código final.

A seguir, apresentamos os principais motores de renderização utilizados no mercado.

1.6.1.1. WebKit

É um motor de renderização criado pela Apple, utilizado inicialmente no Safari, conservado como um projeto de código aberto, é escrito em linguagem C++ e hoje é mantido por um grande grupo de desenvolvedores.

É importante mencionar que existem vários tipos de WebKit e o fato de dois navegadores utilizarem WebKit como motor de renderização não necessariamente os torna idênticos em seu de resultado final.

É utilizado nos seguintes navegadores:

- Safari;
- Google Chrome (além de WebKit, utiliza o Google V8 JavaScript Engine). Em abril de 2013, o Google anuncia que está desenvolvendo seu próprio motor de renderização, denominado Blink e que será adotado em seu navegador Google Chrome;
- Konqueror;
- Opera (Em 2013, o Opera mudou de Presto para WebKit e também irá utilizar o Google V8 JavaScript Engine). Em abril de 2013, o Opera anuncia que também mudará para o Blink, projeto de código aberto criado pelo Google. A ideia é o aumento de desempenho e o multiprocessamento.

1.6.1.2. Gecko

É um motor de renderização criado pela Fundação Mozilla. Inicialmente chamado de NGLayout.

Embora seja mantido pela Fundação Mozilla, Gecko é uma marca registrada da Netscape Communications Corporation, empresa pertencente ao grupo AOL.

É utilizado nos seguintes navegadores:

- Firefox;
- Mozilla Suite;
- Camino;
- Flock;
- K-Meleon;
- Thunderbird (e-mail).

1.6.1.3. Trident

É um motor de renderização proprietário da Microsoft e embora seja um dos primeiros a implantar recursos de formatação de estilos, acabou por gerar inúmeros problemas de incompatibilidade.

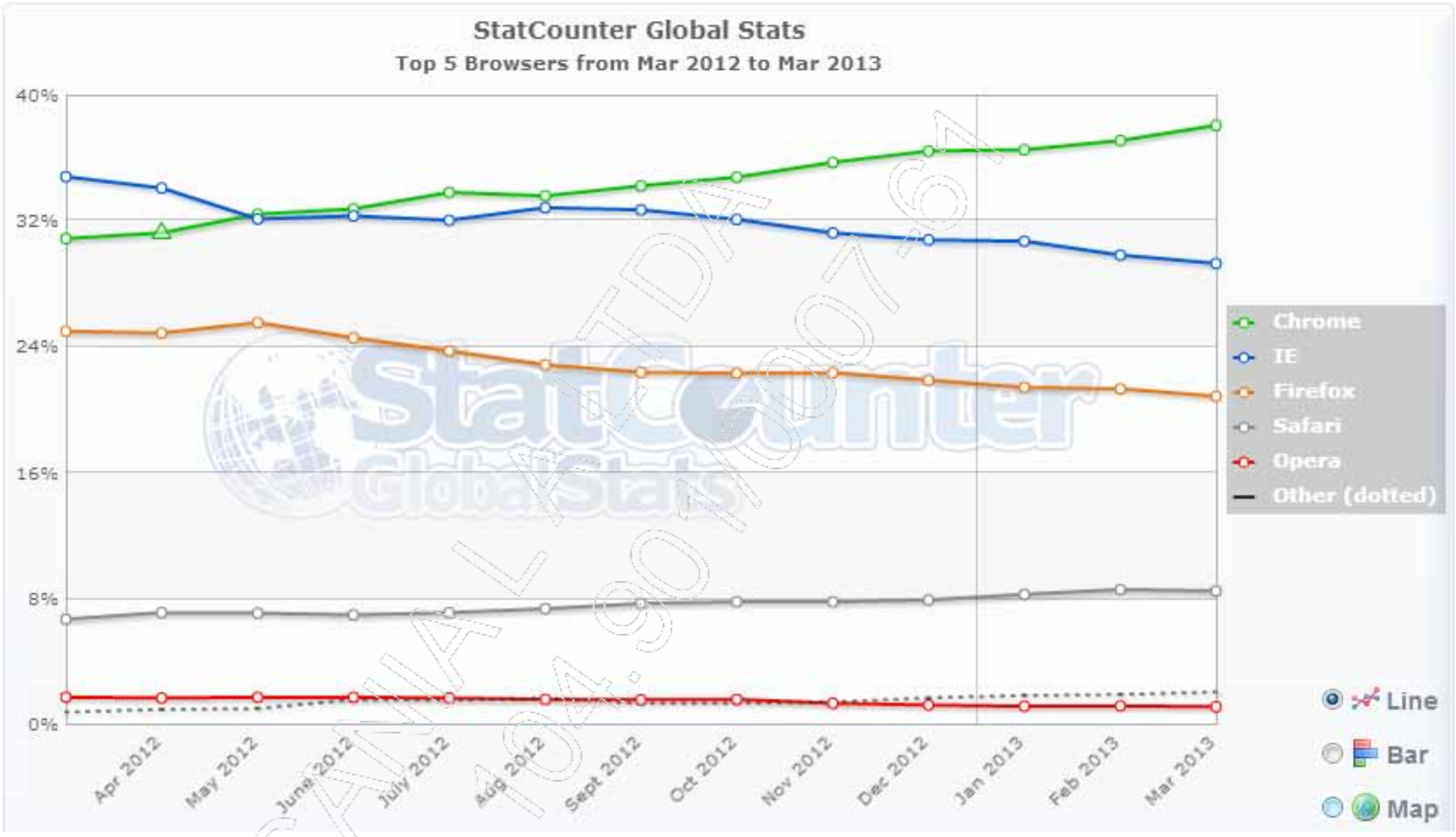
Cada um dos navegadores Internet Explorer de versões 6 a 8 interpretava o código à sua maneira, o que tornava o trabalho do profissional responsável pelo layout um tanto quanto maçante, uma vez que um mesmo site tinha várias formas de comportamento para um mesmo mecanismo. Com a versão 6.0 do Trident utilizada no Internet Explorer 10, a tentativa da Microsoft é manter uma maior compatibilidade com o Padrão Web (Web Standard).

É utilizado nos seguintes navegadores:

- Internet Explorer;
- Microsoft Outlook (e-mail);
- Visual Studio (IDE).

1.6.2. Principais Navegadores

Segundo as estatísticas de vários sites de pesquisa globais, entre eles o StatCounter Global Stats, em 2013, o Google Chrome lidera a lista dos principais navegadores utilizados no mercado com 38.07% de Market Share (Participação de mercado), seguido pelo Internet Explorer com 29.03% (o que inclui todas as versões do IE), em seguida, Firefox com 20%, Safari com 8% e Opera com 1.17%.



StatCounter Global Stats, disponível em gs.statcounter.com/

Os navegadores Google Chrome e Firefox são atualizados automaticamente a cada seis semanas e possuem uma versão para desenvolvedores, com a intenção de demonstrar novos recursos que serão implementados em futuras versões. O Google Chrome Canary é a versão para desenvolvedor, disponível para download em https://www.google.com.br/chrome/browser/canary.html. O Mozilla Firefox Aurora é a versão do Firefox para desenvolvedor.

Teste seus conhecimentos

Introdução ao HTML5

1

1. O que significa o acrônimo HTML?

- ☐ a) Hypertext Master Language
- ☐ b) Hypertext Master Link
- ☐ c) Hyper-Text Markup Link
- ☐ d) Hypertext Markup Language
- ☐ e) Hyperlink Markup Language

2. Quem criou o HTML?

- ☐ a) Steve Jobs
- ☐ b) Tim Berners-Lee
- ☐ c) Willian Gates
- ☐ d) Steve Wozniak
- ☐ e) Mark Zuckerberg

3. Qual motor de renderização é utilizado no Internet Explorer?

- ☐ a) Nenhum
- ☐ b) WebKit
- ☐ c) Presto
- ☐ d) Gecko
- ☐ e) Trident

4. Qual motor de renderização é utilizado no Mozilla Firefox?

- ☐ a) Samba
- ☐ b) WebKit
- ☐ c) Presto
- ☐ d) Gecko
- ☐ e) Trident

5. Qual motor de renderização é utilizado no Google Chrome?

- ☐ a) MSHTML
- ☐ b) WebKit
- ☐ c) Presto
- ☐ d) Gecko
- ☐ e) Flock

6. Qual motor de renderização é utilizado no Safari?

- ☐ a) V8 JavaScript Engine
- ☐ b) WebKit
- ☐ c) Flock
- ☐ d) Lynx
- ☐ e) Trident

O universo HTML5 2

- ✓ A linguagem HTML5;
- ✓ Logotipo oficial;
- ✓ Tecnologias da HTML5.

2.1.A linguagem HTML5

A linguagem de marcação HTML5 facilita o trabalho de desenvolvedores, tornando possível a manipulação de elementos e a modificação de características dos objetos de maneira leve e funcional. HTML5 possui ferramentas para CSS e JavaScript, bem como cria novas tags e modifica a função de outras. Assim, elementos e atributos que foram modificados podem ser usados de maneira mais eficaz.

Nas versões anteriores do HTML, não existia um padrão para criação de seções comuns e específicas (rodapés, cabeçalhos etc.), como também não havia um padrão de nomenclatura de IDs, classes ou tags.

O HTML5 facilita a maneira como o código é escrito e como as informações são organizadas na página, possibilitando mais interatividade sem a necessidade de utilização de plug-ins e sem perda de performance.

Dois aspectos são relevantes no HTML5:

- **Interoperabilidade**: Facilita a reutilização de informação em novos dispositivos;
- **Retrocompatibilidade**: Compatibiliza sites já existentes com os browsers recentes, ou seja, não é preciso refazer sites para que estes se adaptem a novos conceitos e regras.

2.2.Logotipo oficial

O logotipo oficial do HTML5 é um distintivo que, de forma simples, transmite muito mais do que apenas o envolvimento da tecnologia com os padrões abertos da Web. Encomendado pela W3C, ele foi desenvolvido por uma respeitada agência de publicidade denominada OCUPOP.

Observe o logo:

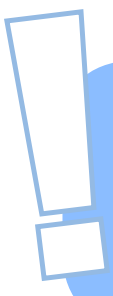
Este logotipo e todos os outros criados pela W3C para HTML5 e suas tecnologias estão licenciados como Creative Commons Attribution 3.0 Unported, ou seja, você pode compartilhar e modificar este logotipo, bem como utilizá-lo em produtos, tais como camisetas, bonés, logotipos estilizados, com o objetivo de difundir a tecnologia. Sempre que possível, você poderá creditar este logotipo com o termo “Criado por W3C”.

2.3.Tecnologias da HTML5

As tecnologias do HTML5 são divididas oficialmente em oito áreas, descritas a seguir.

2.3.1.Semântica

O foco principal da HTML5 é dar sentido ao conteúdo de um site ou de um Web App. Com tecnologias como Microdados e Microformatos, a HTML5 permite que sejam identificados nomes, locais, preferências, empresas em um texto e que essas informações sejam interligadas. Para isso, faz uso de uma rica coleção de tags, dando significado semântico a textos que antes eram apenas conjuntos de palavras sem sentido para as máquinas.



Além de tags semânticas para o conteúdo do texto, a parte de formulários ganhou novas entradas de dados que estão melhores associadas ao seu real objetivo. Há tipos de entrada de dados específicos para e-mail, números, calendários, cores e muito mais.

2.3.2. Offline e armazenamento

Graças a tecnologias como HTML5 App Cache, Local Storage, Indexed DB e as novas especificações de API de arquivos, os Web Apps e sites podem ser mais rápidos e continuar trabalhando mesmo se não houver conexão com a Internet.

2.3.3. Acesso ao dispositivo

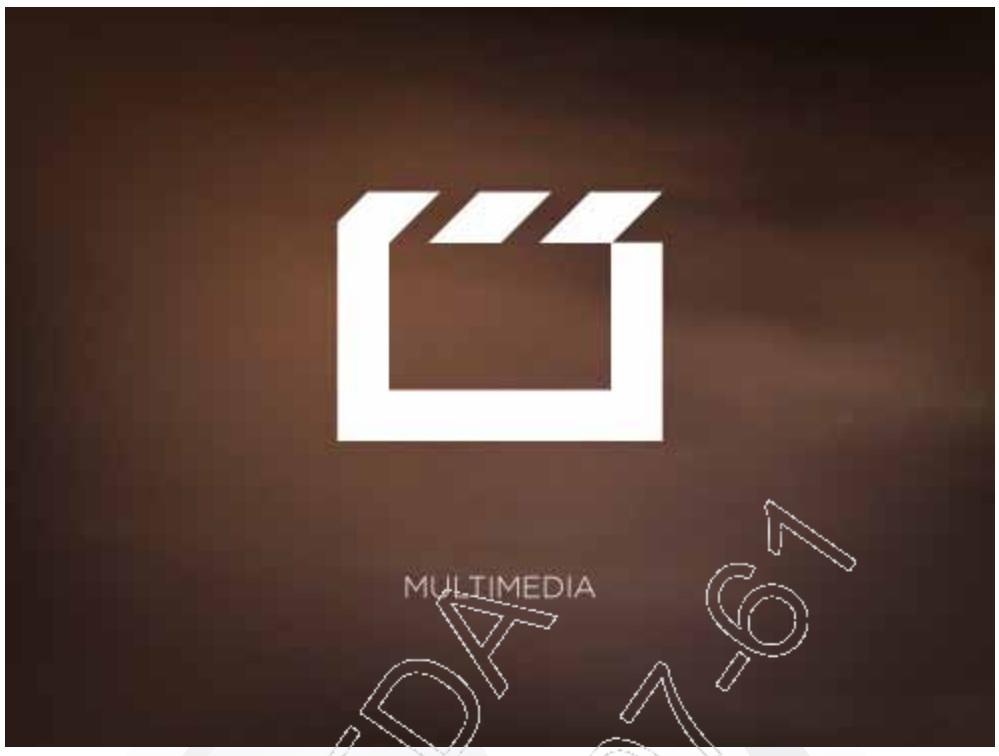
Com a utilização da API de Geolocalização, as aplicações Web podem acessar recursos de dispositivos e melhorar a experiência do usuário: desde áudio/vídeo até microfones e câmeras.

2.3.4. Conectividade

Com mais eficiência em conectividade, que é necessária nas aplicações em tempo real, como chat, temos, também, velocidade maior para jogos e melhor comunicação. O envio de dados entre cliente e servidor também se torna muito mais rápido e eficiente com a utilização de Web Sockets e Server-Sent Events.

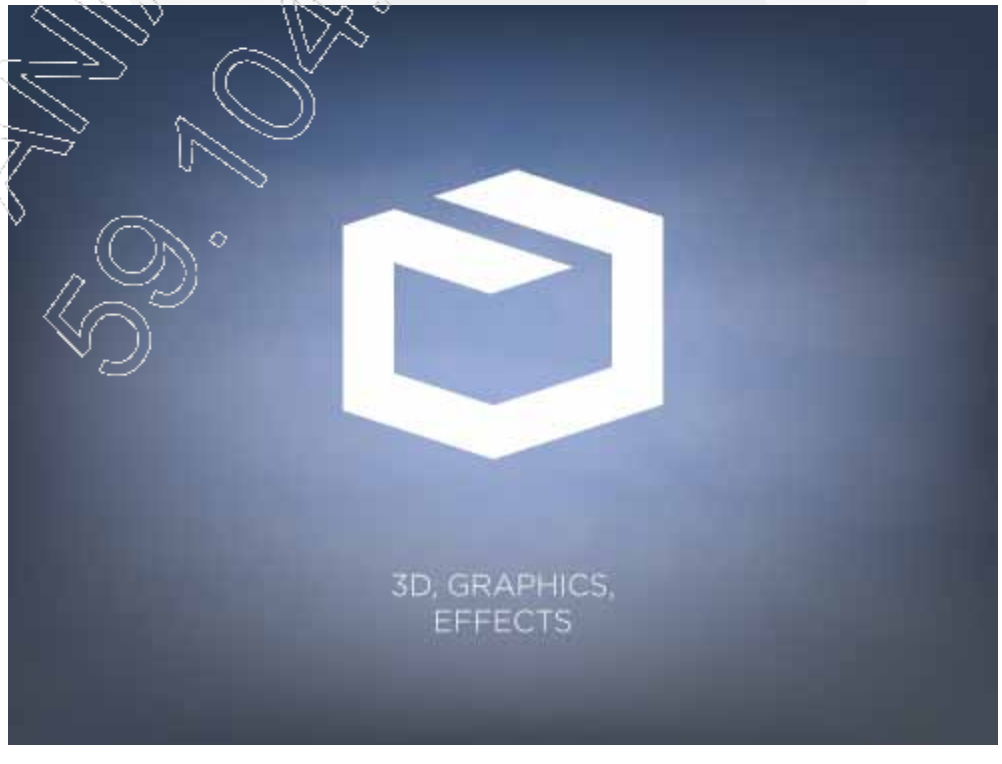
Em Ajax, temos uma requisição cliente/servidor de forma assíncrona. A novidade com Server-Sent Events é que podemos inverter esse processo: o servidor é que envia eventos para o cliente. Isso é ideal para jogos com vários jogadores simultâneos.

2.3.5. Multimídia



Áudio e vídeo são os principais recursos de multimídia para integração com suas aplicações Web e sites.

2.3.6. Gráficos, 3D e efeitos



Com a utilização de SVG, Canvas, WebGL e recursos CSS3 3D, é possível produzir experiências visuais incríveis para o usuário, sem a necessidade de plug-ins, ou seja, nativamente no navegador.

É importante salientar que alguns desses recursos ainda não estão completamente padronizados pela W3C e ainda não foram adotados pelos principais navegadores. Para maiores esclarecimento sobre padronização veja: **www.caniuse.com**.

2.3.7. Desempenho e integração

Utilizando tecnologias como Web Workers e XMLHttpRequest 2, é possível criar suas aplicações Web e conteúdo dinâmico (Ajax) de forma mais rápida, além da grande variedade de técnicas que permite o aumento de desempenho

2.3.8. CSS3

CSS3 é uma das maiores revoluções e, embora seja independente da HTML5, é entendida como uma de suas variantes em termos de novas tecnologias de padrão aberto para estilização e efeitos de um documento formatado. Com ela, você cria suas aplicações Web sem sacrificar a estrutura semântica ou o desempenho. São disponibilizadas novas formas de estilização de fontes, de alinhamento de layout, de seletores avançados, animação e transformação.

Teste seus conhecimentos

O universo HTML5 2

1. Qual categoria do HTML5 é responsável por utilizar o XMLHttpRequest 2?

- ☐ a) Desempenho e acesso ao dispositivo
- ☐ b) Desempenho e integração
- ☐ c) Multimídia
- ☐ d) Desempenho e semântica
- ☐ e) CSS3

2. Qual categoria do HTML5 é responsável por manter a aplicação funcionando mesmo sem conexão com Internet?

- ☐ a) Offline
- ☐ b) Semântica
- ☐ c) Multimídia
- ☐ d) Desempenho e Integração
- ☐ e) Conectividade

3. Quais os principais elementos para a aplicação de multimídia em uma solução Web?

- ☐ a) Áudio e API de áudio.
- ☐ b) Teclado e vídeo.
- ☐ c) Mouse e teclado.
- ☐ d) Áudio e vídeo.
- ☐ e) Áudio e codec.

4. Quais elementos podemos acessar por meio da categoria Acesso ao dispositivo?

- ☐ a) Mouse e eventos touch.
- ☐ b) Monitor, mouse e teclado.
- ☐ c) Microfone, câmera e outros recursos de hardware.
- ☐ d) GPS, acelerômetro e orientação do dispositivo.
- ☐ e) Não é possível acessar o hardware com API de acesso ao dispositivo.

5. Qual a tecnologia responsável por permitir a formatação dos arquivos incluindo fontes, alinhamento, cores e estilização geral do site?

- ☐ a) HTML5
- ☐ b) 3D
- ☐ c) Multimídia
- ☐ d) Semântica
- ☐ e) CSS3

Conhecendo a estrutura HTML5

3

- ✓ Tags e atributos do HTML5;
- ✓ Estrutura do código HTML5;
- ✓ Metatags;
- ✓ Visualização de um código renderizado.

3.1.Tags e atributos do HTML5

Tag é o termo atribuído aos códigos utilizados em HTML. As tags são identificadas por estarem entre o sinal menor que (<) e maior que (>), conforme demonstram os seguintes exemplos: **<html></html>**; **<body></body>**.

A estrutura de um documento HTML é formada basicamente por tags, as quais podem ou não possuir atributos.

É preciso ter em mente que as tags e os atributos que compõem um documento HTML5 devem estar de acordo com algumas regras, a fim de que sua estrutura esteja adequada.

Dentre essas regras, destacamos a necessidade de que um elemento vazio, assim como os outros elementos, tenha uma tag de abertura e outra de fechamento, por exemplo, **<title></title>**. Além disso, os valores referentes aos atributos devem ser colocados entre aspas duplas, até mesmo os valores que são compostos por números.

As tags são hierárquicas, logo, dentro de um par de tags podemos ter outras tags que atuam como elementos filhos das tags principais. É o que ocorre, por exemplo, com a tag **<title></title** que é filha da tag **<head></head>**, que por sua vez é filha do elemento ou tag raiz **<html></html>**.

Veja um exemplo de hierarquia:

```
<html>
        <head>
        <title>HTML5 Fundamentos</title>
        </head>
        <body>
        </body>
</html>
```

Quando essas regras são observadas, o documento HTML5 é construído de forma adequada, reduzindo a possibilidade de que ocorra um problema.

3.2.Estrutura do código HTML5

A seguir, temos um modelo que demonstra a estrutura correta de um código HTML5. Observe:

```
1  <!DOCTYPE html>
2   <html lang=”pt-br”>
3      <head>
4          <meta charset="utf-8">
5          <title>HTML5 Fundamentos</title>
6      </head>
7      <body>
8          Conteúdo
9      </body>
10  </html>
```

Podemos verificar que este exemplo demonstra a estrutura correta de um código HTML, pois todas as regras mencionadas anteriormente foram obedecidas: o valor do atributo foi colocado entre aspas, todas as tags foram declaradas em letra minúscula (caixa baixa) e todas as tags abertas também foram fechadas.

3.2.1.Tipo de documento (DOCTYPE)

DOCTYPE é um tipo de declaração que deve obrigatoriamente constar em um código HTML, sendo incluída antes do elemento raiz desse código. Determina ao navegador qual o tipo de documento ele deverá interpretar.

Diferentemente da linguagem anterior, XHTML, existe agora apenas um tipo de declaração e este é bem simples. Anteriormente, um tipo de declaração do tipo transacional era feito da seguinte maneira:

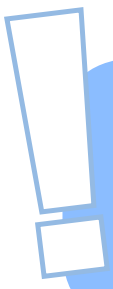
```
<!DOCTYPE html PUBLIC “-//W3C//DTD XHTML 1.0 Transitional//EN” “http://www.
w3.org/TR/xhtml1/DTD/xhtml -transitional.dtd”>
```

Agora, no HTML5, usamos:

```
<!DOCTYPE html>
```

3.2.2. Elemento raiz (html)

O elemento raiz de um código HTML deve ser a tag **<html>.** Tal elemento pode, ainda, possuir um atributo **lang** responsável por determinar qual o idioma do documento em questão, e ainda permitir que os navegadores sugiram a melhor tradução quando algum usuário de um país cujo idioma seja diferente do documento acessar o site. Para o português do Brasil, utilizamos **lang="pt-br"**.



O atributo **lang** pode ser utilizado também dentro de textos no documento com a intenção de determinar o idioma que está sendo utilizado em um determinado local. Por exemplo: ** Bienvenido a HTML5**.

3.2.3. Cabeçalho (head)

O cabeçalho, representado pelo elemento **<head>**, contém informações genéricas a respeito do conteúdo do documento e da forma que será exibido. Essas informações genéricas também são chamadas de metadados, os quais representam informações capazes de descrever outros dados. Portanto, podemos dizer que a função da tag **<head>** é determinar o cabeçalho de uma página.

Todos os elementos que compõem esta tag não são exibidos pelo browser. Dentre esses elementos, podemos ter metatags, folhas de estilo, scripts, entre outros. Dentro da tag **<head>**, deve ser inserida a tag **<title>**, cuja função é determinar o título da página.

Se desejarmos carregar estilos para formatação de conteúdo na própria página, utilizamos o elemento ou tag **<style></style>**. Se os estilos forem carregados externamente, utilizamos o elemento **<link>**.

Quando precisamos adicionar programação de script ao documento, de forma que essa seja carregada antes do corpo do site, podemos adicionar o elemento **<script></script>** como elementos filhos de **<head></head>**.

3.2.4. Corpo da página (body)

O corpo de uma página HTML5 é determinado pela tag **<body>**, cujos elementos têm a função de determinar tudo o que será apresentado pelo browser de forma gráfica. Observe o exemplo a seguir:

```
<body>
Conhecendo a estrutura do código HTML5
</body>
```

3.3. Metatags

Definimos metatags como sendo códigos referentes a informações e a indicações. Assim como as folhas de estilo e os scripts, as metatags representam um dos elementos a serem colocados no cabeçalho da página (tag **<head>**).

O elemento utilizado para realizar a declaração de uma metatag é o **<meta />**, cuja função é fornecer dados capazes de descrever informações presentes no corpo da página. Normalmente, essas informações referem-se à descrição do conteúdo e a palavras-chave.

3.3.1. Meta name

O elemento **<meta name />** costuma referir-se às informações relacionadas ao autor da página, conforme demonstra o exemplo a seguir:

```
<meta name="author" content="autor_da_página" />
```

Podemos observar que **"author"** é um atributo que permite determinar o nome do desenvolvedor da página.

O site **www.humanstxt.org** permite que você crie um arquivo de texto determinando quem são as pessoas por trás do site, e que áreas cada uma delas atuou. Após criá-lo, no elemento **<link />**, é feita uma ligação com esse arquivo de texto. Essa tem sido uma iniciativa bem aceita pela comunidade mundial de desenvolvedores Web, para colocar mais informações sobre os autores do site.

3.3.2. Palavras-chave e descrições

Para compreender a importância das palavras-chave e das descrições, é preciso ter em mente que as buscas dinâmicas realizadas por meio dos motores de busca são auxiliadas, entre outras regras, por metatags.

A fim de realizar uma pesquisa, os motores de busca utilizam regras que verificam os cabeçalhos e o conteúdo das páginas na rede, para encontrar as palavras-chave que estão presentes nas metatags e que são representadas pelo atributo **keywords**. Esses aplicativos são chamados de robots.

Assim que a palavra-chave é encontrada, todas as descrições contendo o atributo **description** que foram encontradas pelas metatags são exibidas em uma página de resultado.

Embora as metatags possam conter várias palavras-chave, as quais devem ser separadas por vírgulas, a descrição requer mais objetividade, visto que um texto de descrição bastante extensa pode não ser visualizado por inteiro, uma vez que o motor de busca limita a quantidade de caracteres a ser exibida:

```
<meta name=”keywords” content=”HTML5, CSS3, Semântica, Canvas” />

<meta name=”description” content=”Nossa empresa esta situada no melhor...” />
```

Os mecanismos que encontram as palavras-chave (keywords) apresentam a descrição do site, a qual está presente no atributo **description**.

Quando trabalhamos com a Internet, temos não apenas as ferramentas de busca, mas também os catálogos, que são sites destinados à realização de pesquisas e cujo banco de dados pode ser alimentado, desde que seja feito um cadastramento para isso.

3.3.3. Meta charset

Quando precisamos tornar nosso site disponível para os principais idiomas, precisamos determinar o encoding que o site utilizará. Existe um encoding que permite utilizar cerca de um milhão de caracteres: chama-se **utf-8**.

Para determinar o encoding utilizado no documento, temos o elemento **meta** com o atributo **charset <meta charset/>**:

```
<meta charset=”utf-8”>
```

3.4.Visualizando o código renderizado

Para visualizar o código renderizado, devemos salvar o arquivo com a extensão **.html**. Vale destacar que visualizar o código renderizado significa visualizar o resultado obtido com os códigos a partir do browser.

Após salvarmos o documento como **.html**, podemos abri-lo de duas formas distintas: com um duplo clique sobre o nome do arquivo salvo, ou a partir do browser utilizado. É importante destacar que, para criar uma página HTML5, podemos utilizar um editor de textos simples.

Existem inúmeros editores de texto e Ambientes Desenvolvimento Integrado (IDE - Integrated Development Enviroment). Veja a lista com alguns dos softwares mais utilizados no mercado:

- Adobe Dreamweaver;
- Sublime Text;
- Aptana Studio;
- Eclipse;
- Vim;
- Notepad++;
- Editplus.

Teste seus conhecimentos

Conhecendo a estrutura HTML5

3

1. Quais as tags principais e em que ordem elas formam a estrutura de um documento HTML válido?

- ☐ a) <html> <form></form><head></head></html>
- ☐ b) <html><body></body><head/>
- ☐ c) <html><head><body></body><title>
- ☐ d) <html><head><title></title></head><body></body></html>
- ☐ e) <html><head><title></title><body></head></html>

2. Qual encoding deve ser utilizado para tornar o documento compatível com os principais idiomas atuais?

- ☐ a) iso8859-1
- ☐ b) version=2.0
- ☐ c) utf-8
- ☐ d) utf16
- ☐ e) lang=”pt-br”

3. Para determinar informações do documento HTML utilizamos a tag meta. Qual atributo deve ser utilizado para apresentar uma descrição do site nos motores de busca?

- ☐ a) search
- ☐ b) description
- ☐ c) name
- ☐ d) humanstxt
- ☐ e) body

4. Para determinar informações do documento HTML utilizamos a tag meta. Qual atributo deve ser utilizado para configurar o encoding da página?

- ☐ a) content
- ☐ b) contenttype
- ☐ c) name
- ☐ d) author
- ☐ e) charset

Mãos à obra!

Conhecendo a estrutura HTML5

3

Laboratório 1

A – Criando um documento HTML com os elementos principais

1. Crie um diretório chamado **labs** e, dentro dele, um arquivo com o nome **lab_aula3.html**;

2. Abra o arquivo no editor de texto e determine o doctype apropriado para HTML5;

3. Crie os elementos principais de um documento HTML5 válido;

4. Determine o título **HTML5 Fundamentos Aula 3**;

5. No conteúdo do documento, insira a frase: **Primeiro laboratório de HTML5***;*

6. Salve o documento e execute no navegador;

7. Clique com o botão direito do mouse sobre o documento e mande exibir o código fonte;

8. Verifique se a estrutura exibida condiz com o que foi criado.

Formatando um documento HTML5

4

- ✓ Entendendo a semântica de um documento;
- ✓ Elementos de um documento HTML5.

4.1.Introdução

Esta leitura tem por objetivo apresentar as tags ou elementos e os atributos que são utilizados com a finalidade de formatar um documento HTML5. Veja, a seguir, detalhes a respeito deste assunto.

4.2.Entendendo a semântica de um documento

Quando criamos um documento HTML5, seja ele um site ou um aplicativo Web (Web App), precisamos conhecer a estrutura dos elementos principais de um documento HTML5 bem formatado.

Em sua nova versão, o HTML permite utilizarmos elementos que possuem significado tanto para os desenvolvedores, quanto para os motores de busca. Isso é chamado de semântica.

Em versões anteriores do HTML, as áreas de um site eram divididas por tags que não possuíam significado semântico; frequentemente, o elemento ** ** era utilizado quando fosse necessário criar um menu para o site era usado uma sintaxe semelhante a imagem a seguir:

```
<div id="menu">
    <a href="#">Home</a> |
    <a href="#">Contato</a>
</div>
```

Agora, com o HTML5, temos novos elementos semânticos, que dão pleno significado ao documento. Em um contexto de menu de navegação, é possível utilizar o elemento **<nav>** criado especificamente como uma seção que contém um menu de navegação:

```
<nav>
    <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">Contato</a></li>
    </ul>
</nav>
```

Além dos elementos de navegação, existem inúmeros elementos semânticos que atendem à demanda das aplicações Web e dos mecanismos de busca.

4.3.Elementos de um documento HTML5

Em um documento HTML5, todos os elementos começam com a definição do tipo do documento. Isso é possível por meio da tag **<!doctype HTML>**, seguida pela abertura da tag **<html>**.

Após essa definição, temos a criação do cabeçalho do documento por meio da tag **<head>**, que irá definir o que deve ser carregado antes da página ser exibida ao usuário.

Definido o cabeçalho e seu título, o corpo do documento é criado por meio da tag **<body>**, formando a estrutura básica a seguir:

```
1  <!doctype HTML>
2  <html>
3      <head>
4          <title>HTML5 Fundamentos</title>
5      </head>
6      <body>
7
8
9      </body>
10 </html>
```

Dentro da tag **body**, colocamos o conteúdo de nosso site, que será dividido por tags que representam seu significado.

Veja, a seguir, uma lista com os principais elementos do HTML5 e seus respectivos significados:

- **html**

Este é o elemento raiz do documento HTML, ficando logo no início. Dentro dele, ficam todos os elementos filhos.

- **head**

Este é o primeiro elemento filho após o elemento raiz; é o cabeçalho do documento, que deve ser carregado antes da página ser exibida ao usuário. Dentro dele, utilizamos a tag **<meta charset=”utf-8” />**.

- **link**

Elemento responsável por criar uma referência para um arquivo externo, possui um atributo **rel** (relation) para explicar o motivo dessa referência. O valor mais comum para **rel** é **Stylesheet**, referindo-se a um arquivo externo que irá dar forma e organização ao documento atual, formatando cor, fontes, formas e todo o documento.

- **body**

O início do corpo do documento.

- **header**

O cabeçalho do documento. Quando o elemento **header** for filho de outro elemento dentro de um documento, por exemplo, dentro de uma **section**, ele será o cabeçalho deste bloco específico. Logo, em um documento podemos ter vários blocos **section** e cada um ter seu **header** independente.

- **main**

O conteúdo principal da página é alocado dentro da tag **main**. É recomendável utilizar o atributo **role="main"**, uma vez que nem todos os navegadores implementaram esse novo elemento. Informações não essenciais podem ser deixadas de fora.

- **section**

Uma seção com conteúdos diversos no documento, por exemplo, a introdução ou o título. Um layout pode ser dividido em várias sections, desde que possuam conteúdo que reflitam algum sentido semântico.

- **nav**

Uma seção contendo um menu de navegação, formado por víclos que ligam um documento ou página a outra página.

- **article**

Quando é necessário armazenar o texto principal do documento, por exemplo, um artigo de uma revista, jornal, ou blog, ele deve ficar dentro do elemento **article**, que pode possuir um cabeçalho com o título principal e subtítulo. É recomendável a criação de apenas um elemento **article** por documento.

- **aside**

Uma seção que contém conteúdo relacionado aos elementos em sua volta. É comumente utilizada para barras laterais, tais como publicidade, podendo conter outros elementos filhos como **nav**, representando um menu de opções.

- **h1- h6 e hgroup**

Estes elementos permitem criar o título principal do documento utilizando **h1** e os subtítulos hierárquicos de **h2** até **h6**. Quando necessário, é possível agrupar esses títulos dentro de um grupo representado pelo elemento **hgroup**.

- **footer**

O rodapé do documento ou de uma **section** pode conter outros elementos filhos, tais como **nav**, tornando-o assim um menu de navegação no rodapé.

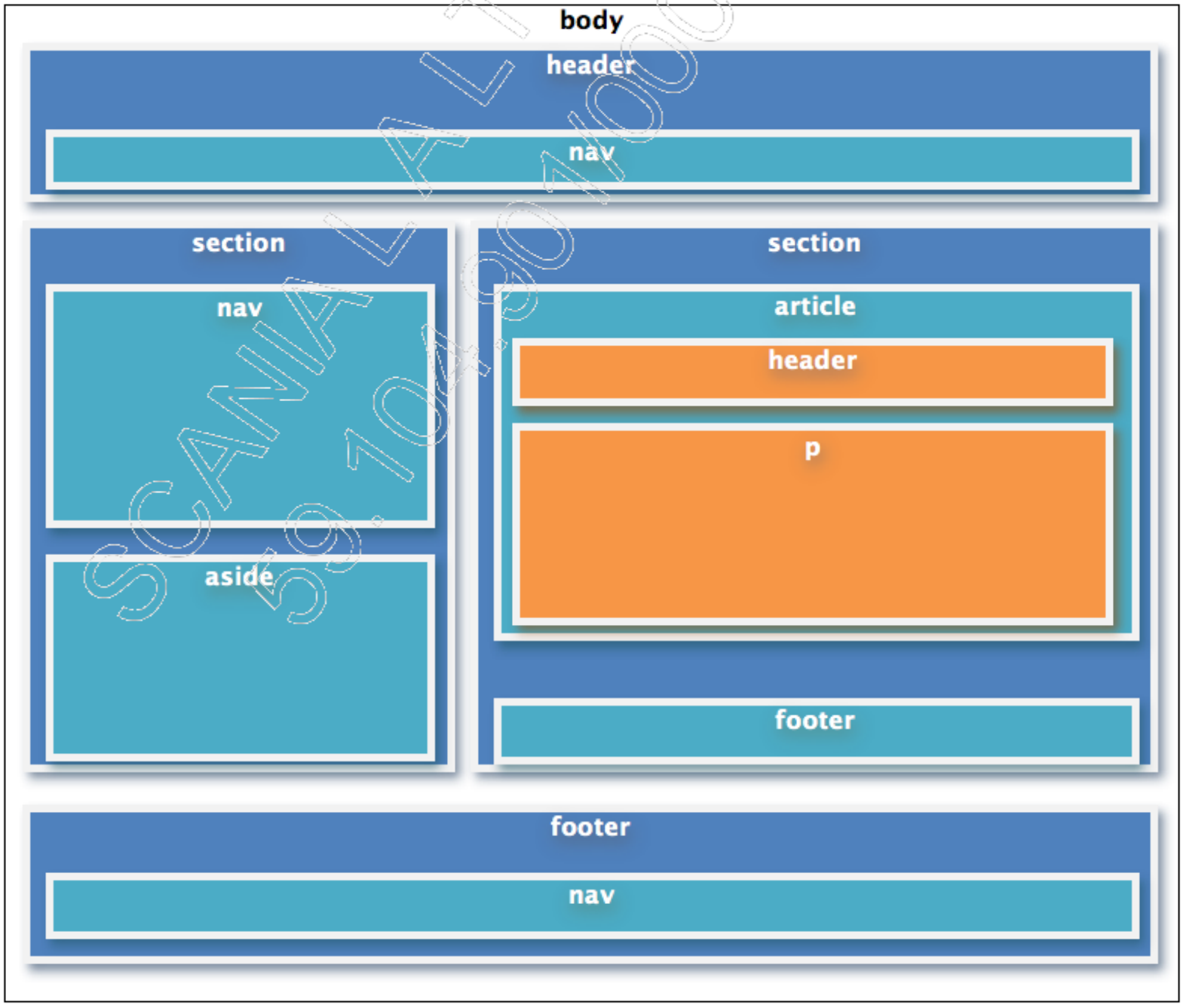
- **p**

Elemento utilizado para páragrafo.

- **div**

Uma seção do site que não possui significado semântico.

Na imagem a seguir, temos a estrutura de um documento, simples e resumida, para mostrar como são estruturados os elementos anteriormente citados. Essa imagem resumida é criada quando o **design de interface** está sendo desenvolvido e a esse recurso damos o nome de **wireframe**. Observe:



Para criar esta estrurura, o procedimento é o seguinte:

```
1  <!doctype HTML>
2  <html>
3      <head>
4          <meta charset="utf-8" />
5          <title>HTML5 Fundamentos</title>
6      </head>
7      <body>
```

O primeiro passo é criar o doctype, o elemento raiz, o cabeçalho do documento, a codificação para acentuação, o título do site, finalizar o cabeçalho e iniciar o corpo do documento:

```
8          <header>
9              <h1>Nome da Empresa</h1>
10             <nav>
11                 <ul>
12                     <li><a href="#">Home</a></li>
13                     <li><a href="#">Loja</a></li>
14                     <li><a href="#">Fale-Conosco</a></li>
15                 </ul>
16             </nav>
17         </header>
```

Agora, criamos o cabeçalho do documento, com o elemento **<header>**. Dentro do dele, colocamos o elemento de título principal **<h1>**, também o elemento de navegação **<nav>**, com uma lista **<ul>**:

```
18         <section>
19             <nav>
20                 <ul>
21                     <li><a href="#">Link lateral</a></li>
22                 </ul>
23             </nav>
24             <aside>
25                 publicidade lateral
26             </aside>
27         </section>
```

Uma seção para os itens laterais é criada, com um menu de navegação utilizando o elemento **<nav>.** Dentro desta seção, é criada uma lista com o elemento **<ul>** e um elemento **<aside>** com informações relacionadas ao conteúdo é adicionado dentro da **section**; este elemento pode representar, por exemplo, uma publicidade:

```
28          <section>
29              <article>
30                  <header>
31                      <h1>Cabeçalho do artigo</h1>
32                  </header>
33                  <p>
34                      texto principal do artigo no site
35                  </p>
36                  <footer>
37                      rodapé do bloco
38                  </footer>
39              </article>
40          </section>
```

Um elemento **<section>** é adicionado para o conteúdo principal e, dentro deste elemento, criamos um elemento **<article>** para o texto principal desta página, com um cabeçalho, um parágrafo e um rodapé.

Quando um elemento **<article>** é criado, os elementos filhos devem fazer referência direta ao texto. Por exemplo, dentro do elemento **<article>**, temos um **<header>** com um **<h1>** para o título do texto principal:

```
41          <footer>
42              Rodapé do site
43              <nav>
44                  <ul>
45                      <li><a href="#">Home</a></li>
46                  </ul>
47              </nav>
48          </footer>
49      </body>
50  </html>
```

Na parte final do documento, é criado um elemento **<footer>**, que determina o rodapé do documento, com uma frase que pode compreender os direitos autorais e um menu de navegação com o elemento **<nav>**, que compreende alguns links importantes do documento:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos</title>
    </head>
    <body>
        <header>
            <h1>Nome da Empresa</h1>
            <nav>
                <ul>
                    <li><a href="#">Home</a></li>
                    <li><a href="#">Loja</a></li>
                    <li><a href="#">Fale-Conosco</a></li>
                </ul>
            </nav>
        </header>
        <section>
            <nav>
                <ul>
                    <li><a href="#">Link lateral</a></li>
                </ul>
            </nav>
            <aside>
                publicidade lateral
            </aside>
        </section>
        <section>
            <article>
                <header>
                    <h1>Cabeçalho do artigo</h1>
                </header>
```

```
33                  <p>
34                      texto principal do artigo no site
35                  </p>
36                  <footer>
37                      rodapé do bloco
38                  </footer>
39              </article>
40          </section>
41          <footer>
42              Rodapé do site
43              <nav>
44                  <ul>
45                      <li><a href="#">Home</a></li>
46                  </ul>
47              </nav>
48          </footer>
49      </body>
50  </html>
```

Com o documento completo, a importância dos elementos semânticos do HTML5 é reforçada, cada parte da página possui um elemento que dá sentido ao conteúdo e organiza as informações tanto para o usuário quanto para os mecanismos de busca.

Teste seus conhecimentos

Formatando um documento HTML5

4

1. O que significa semântica no HTML5?

- ☐ a) Um recurso para criar formulários rapidamente.
- ☐ b) Dar sentido e significado ao código criado.
- ☐ c) Uma forma de separar o que é texto do que é imagem.
- ☐ d) Um recurso que facilita a compreensão para os motores de busca.
- ☐ e) As alternativas “b” e “d” estão corretas.

2. Qual elemento é responsável por criar uma ou mais áreas no site com conteúdo semanticamente relevante?

- ☐ a) Section
- ☐ b) Header
- ☐ c) Head
- ☐ d) Link
- ☐ e) Body

3. Qual a diferença entre os elementos article e section?

- ☐ a) Um article pode conter vários section.
- ☐ b) O elemento article é utilizado para imagens e section para textos.
- ☐ c) O elemento section é uma seção do site, podendo existir vários, enquanto article é para o texto principal, como uma matéria de revista ou blog.
- ☐ d) Um elemento article é uma seção do site, podendo existir vários, já section podemos ter apenas um.
- ☐ e) Os elementos article e section são iguais.

4. Podemos utilizar o elemento header para qual objetivo?

- ☐ a) Colocar o cabeçalho do documento, incluindo um menu de navegação.
- ☐ b) Colocar o cabeçalho de um bloco, por exemplo, de uma seção.
- ☐ c) Colocar o título do site e o elemento nav.
- ☐ d) As alternativas a, b, c estão corretas.
- ☐ e) Nenhuma das alternativas anteriores está correta.

5. Quando desejamos criar um conteúdo opcional que ficará na lateral do documento, qual elemento é mais apropriado semanticamente?

- ☐ a) div
- ☐ b) aside
- ☐ c) section
- ☐ d) sidebar
- ☐ e) article

Mãos à obra!

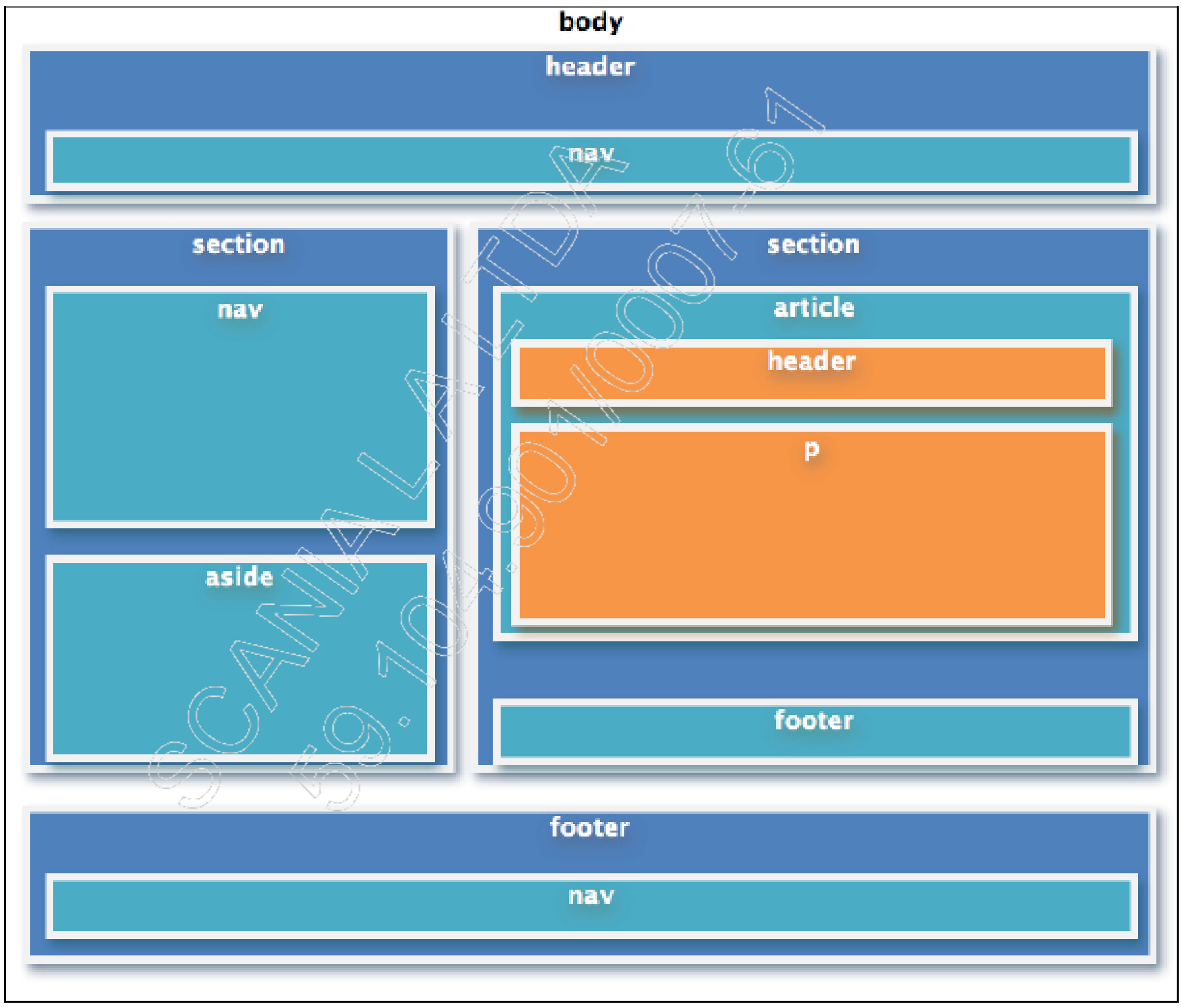
Formatando um documento HTML5

4

Laboratório 1

A – Criando um código HTML5 formatado e estruturado

1. Dado o wireframe a seguir, crie um código HTML5 que represente a estrutura deste código:



Imagem, áudio e vídeo

5

- ✓ Imagem;
- ✓ Áudio;
- ✓ Vídeo.

5.1.Introdução

Apresentaremos, nesta leitura, os elementos **imagem, audio** e **video**, bem como a inserção de mídia em documento HTML5 e as opções para alguns tipos de arquivos.

Os elementos multimídia têm o objetivo de enriquecer a formatação do conteúdo. São utilizados em logotipos, banners de publicidade, ícones, botões, vídeos corporativos, clips, músicas, filmes etc.

5.2.Imagem

As imagens são importantes não apenas para a estética de um documento HTML5, como também para a transmissão de mensagens mais objetivas ao público. Um dos aspectos mais importantes sobre imagens na Web se refere ao peso de uma imagem.

Desde o princípio, em desenvolvimento Web, deve-se pensar no desempenho do site ou aplicativo Web: imagens com boa resolução não raro aumentam o peso do arquivo. Enquanto um arquivo de 1 ou 2 MB pareça pequeno num sistema operacional, quando se trata da Internet é um peso muito grande, que levará tempo para ser carregado.

Utilizando softwares editores de imagem, é possível reduzir o tamanho e o peso de arquivos sem comprometer a qualidade do mesmo para o uso na internet.

Arquivos mais leves permitem que um documento seja aberto em menos tempo.

Devemos levar em conta que muitos usuários utilizam a Internet por meio de dispositivos móveis, embora neste caso seja possível criar um mecanismo que detecte a origem do acesso do usuário, carregando imagens mais leves para celulares e tablets. O ideal é que todo o site possua conteúdo leve sempre que possível.

Os arquivos de imagens utilizados em documentos HTML5 podem possuir as extensões JPG, PNG e GIF. São utilizados basicamente nas seguintes situações:

- **JPG**: Padrão para utilização de imagens, mantém excelente qualidade. Seu peso depende da qualidade e dimensão da imagem;

- **GIF**: Utilizado para pequenas imagens, pequenas animações;

- **PNG**: Utilizado quando precisamos fazer uso de transparência em imagem. Embora o formato **GIF**, também permita transparência, a qualidade do PNG é muito superior;

- **WebP**: Um novo formato de imagem criado e anunciado pelo Google em 2013. Mantém a mesma qualidade de uma arquivo JPG, porém, com 31% de redução no peso e, além disso, possui recurso de transparência e pode ser animado;

- **BMP**: Não recomendamos o uso de imagens BMP por serem muito pesadas;

- **SVG**: Um novo formato de imagem vetorial, permite redimensionar a imagem sem perder a resolução.

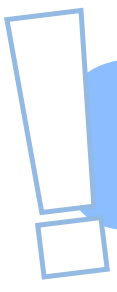
Embora o formato SVG seja o menor, por ser um código estruturado em XML, o formato JPG é o mais utilizado na Web, seguido pelo PNG.

5.2.1.O Elemento IMG

Para que uma imagem seja inserida em um documento HTML5, devemos utilizar o elemento ou tag **<img />**, além de alguns atributos que formam a chamada da imagem.

Além da tag **<img />**, será preciso incluir uma breve descrição da imagem que será colocada no documento. Essa descrição deve ser feita por meio de um atributo chamado **alt** (uma abreviação para alternate - texto alternativo) que, mesmo que vazio, sempre deve ser colocado.

Junto a essa tag, também devemos inserir um **URL** que definirá qual será o arquivo de imagem utilizado.



URL é um acrônimo para **Uniform Resource Locator**, ou seja, um endereço referente a um arquivo ou um documento disponível na rede.

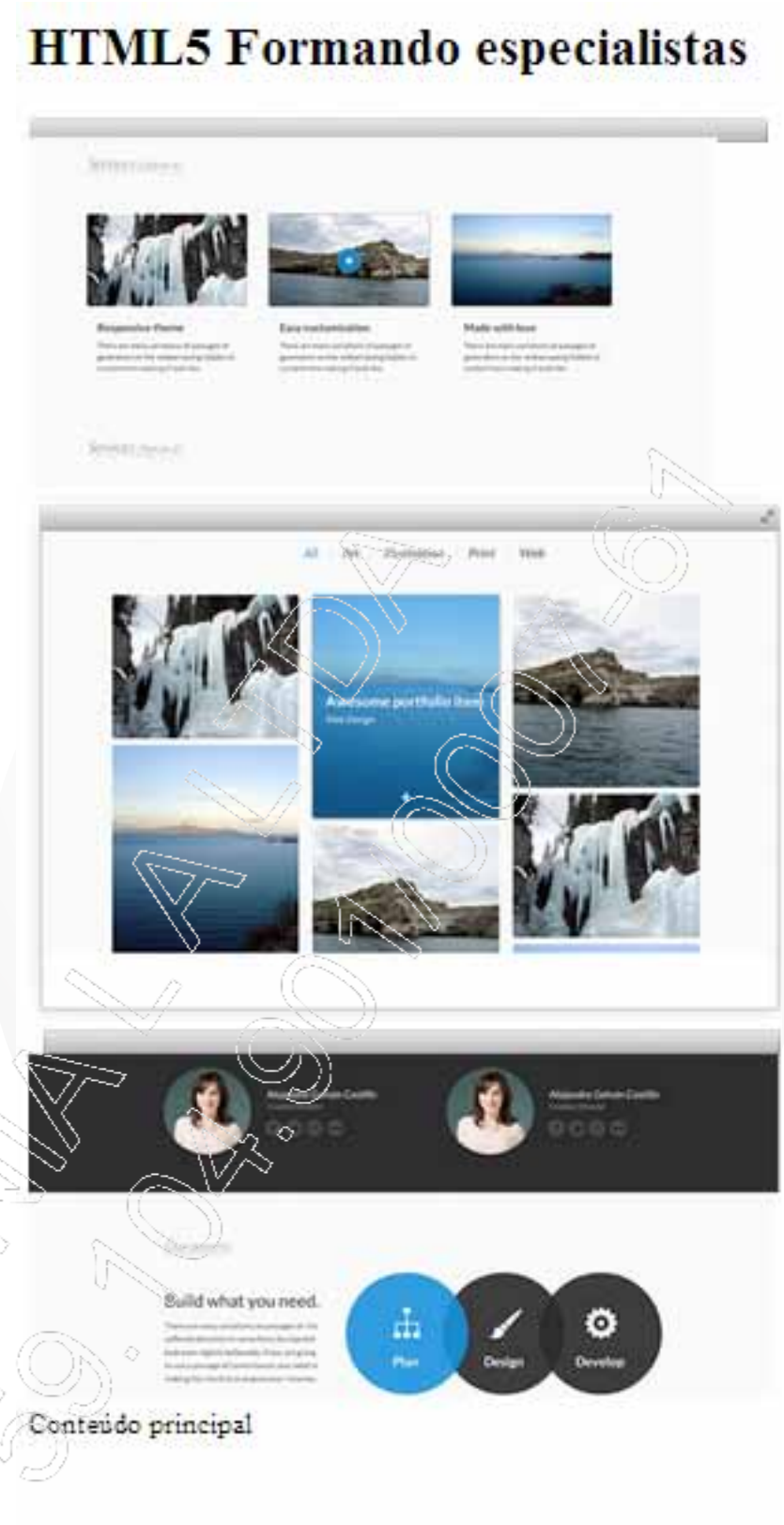
Para representar o valor URL de uma imagem, devemos utilizar o atributo **src**. No exemplo a seguir, podemos verificar a aplicação de ambos os atributos, **alt** e **src**, junto à tag **<img />**:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Imagens</title>
    </head>
    <body>
        <header>
            <h1>HTML5 Formando especialistas</h1>
            <img src="images/left.png" alt="imagem esquerda" class="left"/>
            <br/>
            <img src="images/middle.png" alt="imagem central" class="middle"/>
            <br/>
            <img src="images/right.png" alt="imagem direita" class="right"/>
        </header>
        <main role="main">
            Conteúdo principal
        </main>
    </body>
</html>
```

O exemplo anterior carrega três imagens com extensão **.png**, colocando uma abaixo da outra, com o elemento **
** responsável por quebrar a linha.

Percebemos também que o elemento **<img/>** possui o atributo **src** com a localização do arquivo, o atributo **alt** com a descrição da imagem e um atributo **class**, para formatação com as folhas de estilo em cascata.

O resultado é a imagem a seguir:



Conteúdo principal

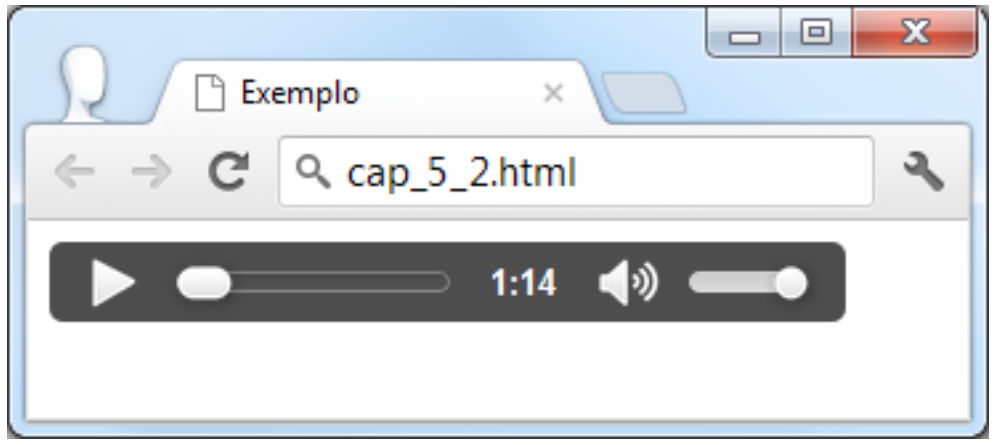
Com a formatação CSS3, as imagens anteriores e uma imagem de fundo mudam completamente o resultado. Veja o resultado com a aplicação de folhas de estilos:

5.3.audio

O elemento **audio** é usado para inserção de som ou stream de áudio em um documento HTML5, sem necessidade de plug-ins, Flash, QuickTime etc. Veja o exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <audio
        controls="controls"
        src="resources/Pirates_of_the_Caribbean.mp3">
    </audio>

</body>
</html>
```

Esse código produzirá o seguinte resultado:



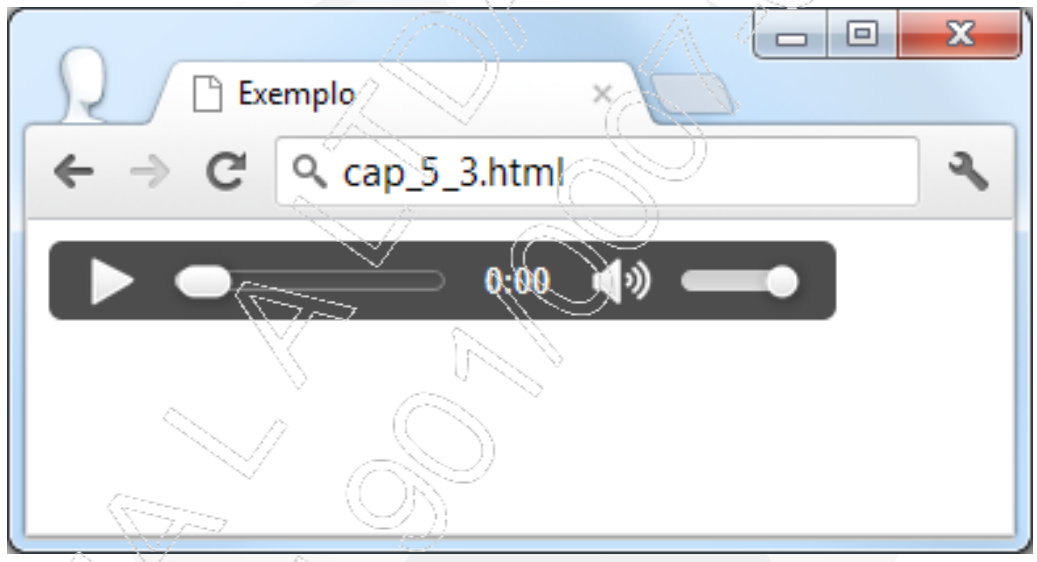
Ao usar o atributo **src** do elemento, indica-se o caminho para o arquivo de som, que é reproduzido pelo navegador. O atributo **controls** indica que os controles Play, Pause, Volume, Mute e Temporizador devem estar visíveis.

O conteúdo do elemento **audio** é renderizado por navegadores que suportam esse elemento. Assim como fazemos em (X)HTML, aqui também definimos a incorporação do som, por exemplo, com o uso de Flash. Veja:

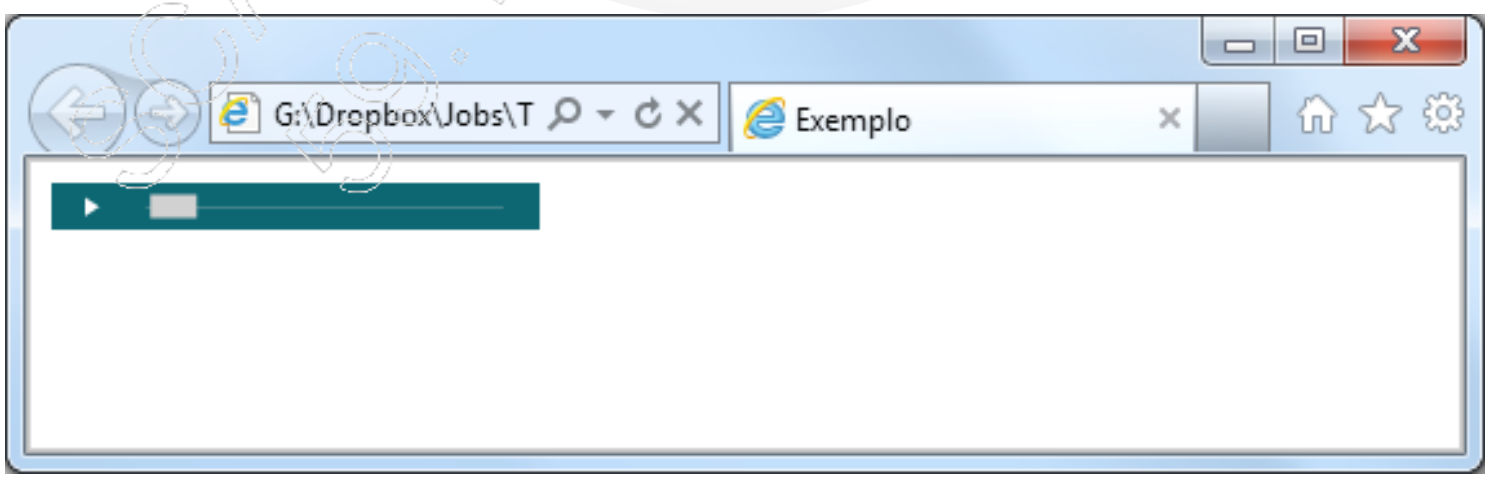
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <audio src="resources/Pirates_of_the_Caribbean.mp3" controls>
        <object
          class="playerpreview"
          type="application/x-shockwave-flash"
          data="resources/audio.swf"
          width="200"
          height="20">
        <param
          name="movie"
          value="resources/audio.swf" />
        <param
          name="bgcolor"
          value="#085c68" />
        <param
          name="FlashVars"
          value="mp3=resources/Pirates_of_the_Caribbean.mp3" />
```

```
25            <embed
26              href="resources/audio.swf"
27              bgcolor="#085c68"
28              width="200"
29              height="20"
30              name="movie"
31              type="application/x-shockwave-flash"
32              flashvars="mp3=resources/Pirates_of_the_Caribbean.mp3" />
33          </object>
34        </audio>
35
36  </body>
37  </html>
```

O uso da tag **audio**, no Google Chrome, será desta forma:



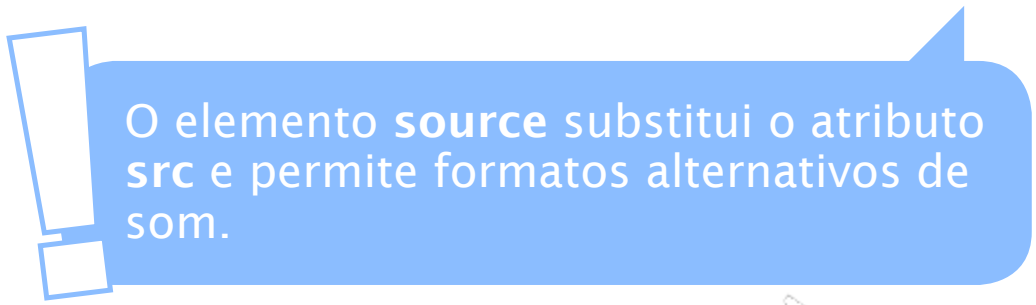
Já o uso do Flash no Internet Explorer 8 será da seguinte maneira:



Segundo o exemplo, quando o navegador suporta HTML5, ele simplesmente ignora o código de inserção de som com Flash, caso contrário, ou seja, quando o navegador não suporta HTML5, ele executa o código Flash.

Se você pretende criar uma implementação crossbrowser, deve inserir marcações para os formatos MP3 e Ogg Vorbis, uma vez que não existe um consenso sobre a adoção de **codecs** para implementação nativa.

No elemento **audio**, você pode usar o elemento **source** como elemento-filho. Ele permite a definição de arquivos diferentes para incorporação de mídias de áudio e vídeo na página.



Veja na tabela a seguir os atributos que o elemento **source** admite e as respectivas características:

Atributo	Característica
src	Indica o endereço do arquivo de mídia a ser incorporado na página. É obrigatório e seu valor é URL não vazio.
type	Indica ao navegador o tipo de mídia a ser incorporado. Seu valor deve ser um **MIME type** válido. Também admite o parâmetro **codecs** definido por **MIME types**, o qual deve ser informado para que o navegador identifique como o arquivo foi codificado.
media	Indica o tipo de mídia. Seus valores possíveis estão listados nas especificações para Media Queries e seu valor-padrão é **all**.

5.4.video

O elemento **video** é usado para inserção de vídeo ou filme em um documento HTML5.

Você deve utilizar alguns atributos desse elemento antes de publicá-lo, porém não há necessidade de plug-ins, Flash, Windows Media Player etc. Veja o exemplo a seguir:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <video controls src="resources/Pirates_of_the_Caribbean.mp4"></video>
10
11 </body>
12 </html>
```

Ao usar o atributo **src** do elemento, indica-se o caminho para o arquivo de vídeo, que é reproduzido pelo navegador.

O conteúdo do elemento **video** é renderizado por navegadores que não suportam esse elemento. Assim como fazemos em (X)HTML, aqui também definimos a incorporação do vídeo, por exemplo, com o uso de Flash. Veja:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <video
        controls
        poster="resources/Pirates_of_the_Caribbean.jpg"
        src="resources/Pirates_of_the_Caribbean.mp4"
        width="480" height="270">
        <object type="application/x-shockwave-flash" data="resources/video.swf"
        width="480" height="270">
            <param name="allowfullscreen" value="true">
            <param name="allowscriptaccess" value="always">
            <param name="flashvars" value="file=resources/Pirates_of_the_Caribbean.mp4">
            <!--[if IE]><!-->
            <param name="movie" value="resources/video.swf">
            <!--<![endif]-->
            <img src="resources/Pirates_of_the_Caribbean.jpg"
                width="480" height="270" alt="Video">
        </object>
    </video>

</body>
</html>
```

Agora dê uma olhada no resultado:

3.1. Trailer oficial do filme Pirates of the Caribbean 4 – On Stranger Tides, Walt Disney Pictures, 2011.

Segundo o exemplo, quando o navegador suporta HTML5, ele simplesmente ignora o código de inserção de vídeo com Flash, caso contrário, ou seja, quando o navegador não suporta HTML5, ele executa o código Flash.

Com relação à implementação das funcionalidades de vídeo, não existe um consenso sobre a adoção de um formato de vídeo para implementação nativa.

A seguir, apresentaremos alguns conceitos básicos sobre vídeos na Web:

- As extensões de vídeo, como **.avi**, **.mov**, **.mp4**, indicam que se trata de um arquivo de vídeo, bem como dão informações sobre a compressão do arquivo. Por exemplo, um arquivo **.mp4** é um contêiner de compressão de vídeo, em que a compressão usada é mp4;

- Um vídeo contém várias informações necessárias para sua apresentação, as quais podem ser trilhas de áudio, stream de vídeo, parâmetros de sincronização etc. Essas informações são comprimidas de várias maneiras, dependendo do arquivo de compressão;

- Além disso, por não haver padronização, os arquivos de vídeo também são codificados de diferentes formas. Essa falta de padronização tanto de contêiners como de **codecs** é decorrente de diferentes implementações dos fabricantes de software em seus produtos;

- Se você pretende criar uma implementação crossbrowser, deve inserir marcações para os vários formatos, uma vez que não existe um consenso sobre a adoção de **codecs** para implementação nativa.

No elemento **video**, você pode usar o elemento **source** como elemento-filho. Ele permite a definição de arquivos diferentes para incorporação de mídias de áudio e vídeo na página.

O elemento **source** substitui o atributo **src** e permite formatos alternativos de vídeo.

Veja na tabela a seguir os atributos que o elemento **source** admite e respectivas características:

Atributo	Característica
src	Indica o endereço do arquivo de mídia a ser incorporado na página. É obrigatório e seu valor é URL não vazio.
type	Indica ao navegador o tipo de mídia a ser incorporado. Seu valor deve ser um **MIME type** válido. Também admite o parâmetro **codecs** definido por **MIME types**, o qual deve ser informado para que o navegador identifique como o arquivo foi codificado.
media	Indica o tipo de mídia. Seus valores possíveis estão listados nas especificações para Media Queries, mas seu valor-padrão é **all**.

O elemento **source** admite também atributos globais, incluindo **width** e **height**.

Teste seus conhecimentos

Imagem, áudio e vídeo

5

1. Qual atributo devemos utilizar para determinar o caminho de uma imagem no HTML5?

- ☐ a) <media/>
- ☐ b) <images>
- ☐ c) src
- ☐ d) alt
- ☐ e) class

2. Qual elemento utilizamos para carregar imagens e inseri-las em um documento HTML5?

- ☐ a) alt
- ☐ b) img
- ☐ c) src
- ☐ d) URL
- ☐ e)

3. Qual o procedimento para carregar um áudio em um documento HTML5 incluindo seu player?

- ☐ a) Utilizar o elemento <audio> com o atributo controls.
- ☐ b) Utilizar o elemento media com o atributo player.
- ☐ c) Utilizar o elemento audio com o atributo player=”true”.
- ☐ d) Utilizar um elemento de vídeo sem imagem.
- ☐ e) Incluir um player do Flash.

4. Qual outro atributo, além de src, podemos colocar no elemento vídeo para determinar o caminho do vídeo?

- ☐ a) alt
- ☐ b) href
- ☐ c) tag
- ☐ d) link
- ☐ e) source

Mãos à obra!

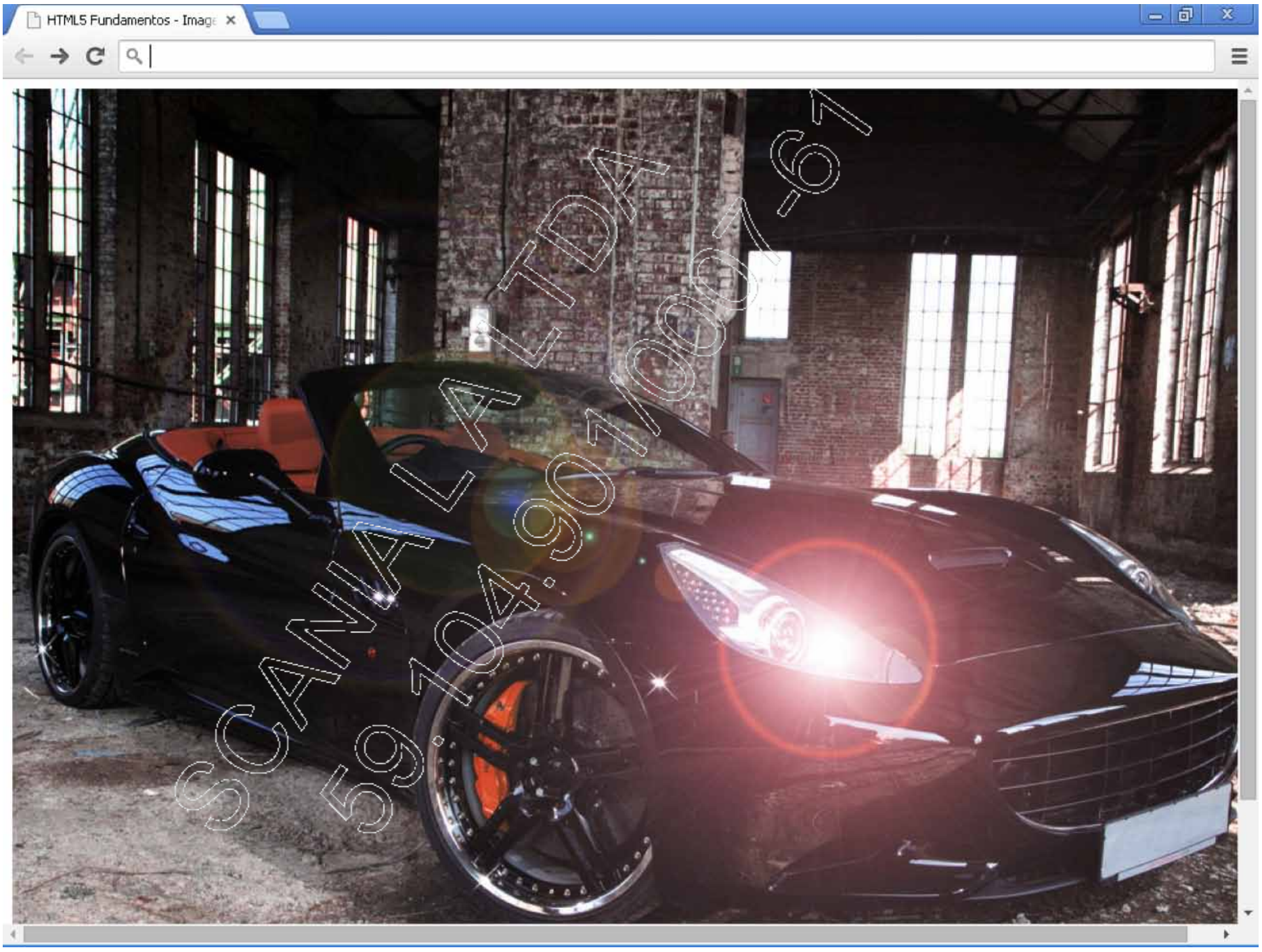
Imagem, áudio e vídeo

5

Laboratório 1

A – Criando um código HTML5 com um elemento imagem

1. Considerando a imagem a seguir e o arquivo **ferrari.jpg,** construa um arquivo chamado **aula5_lab1.html**, carregando a imagem **images/ferrari.jpg** dentro de uma estrutura HTML5, com o texto alternativo **Ferrari Califórnia**.



Trabalhando com vínculos (links) e microdata

6

- ✓ Definição de âncoras;
- ✓ Tipos de vínculos (links);
- ✓ Definição da janela de destino;
- ✓ Arquivos de download;
- ✓ Microdata.

6.1.Introdução

O hipertexto é considerado um dos mais significativos recursos da Internet, pois a partir de suas propriedades, é possível navegar de um documento para outro, em questão de segundos.

Por meio do hipertexto, é possível acessar diferentes tipos de arquivos, como imagens, áudio e texto, de forma rápida e fácil. Para isso, é necessário realizar a formatação correta do código referente ao destino do vínculo (link) que será criado.

Nesta leitura, aprenderemos como trabalhar com essa formatação para viabilizar a navegação entre arquivos a partir de uma página da Internet. Também entenderemos como o advento do microdata pode criar vínculos de informações dentro de um contexto com referências semânticas externas, ajudando a dar sentido a um texto que antes, para os motores de busca, eram apenas caracteres sem qualquer significado.

6.2.Definindo âncoras

O vínculo que permite o acesso a um determinado arquivo é definido através de uma âncora, representada pela tag **<a>**. Por meio de uma âncora, é possível utilizar o atributo **href** para definir um link associado a um arquivo. A âncora permite, ainda, que um bookmark seja criado dentro de um documento, por meio da indicação do atributo **id** ou **name**.

Assim que clicamos em um link associado a uma página, ela poderá ser aberta tanto na janela atual como em outro destino, caso este seja especificado corretamente.

6.3.Tipos de vínculos (links)

As tags **<a>** e **** podem ser utilizadas para estabelecer diferentes tipos de vínculos, cada qual com a sua finalidade, como podemos verificar a seguir.

6.3.1.Link absoluto

Um link absoluto permite que seja definido um vínculo com um endereço completo. O destino deve ser especificado por meio do protocolo **http**, seguido por dois pontos e duas barras (**://**).

No exemplo a seguir, o nome **Impacta Certificação e Treinamento** possui a propriedade de vínculo com um endereço absoluto, já que ela aparece entre as tags **<a>** e ****. Basta o usuário clicar nessa palavra para que ele seja levado ao site da **Impacta Certificação e Treinamento**:

```
<a href="http://www.impacta.com.br" alt="Site da Impacta">
        Impacta Certificação e Treinamento
</a>
```

6.3.2. Link relativo

Um vínculo também pode ser criado para documentos encontrados dentro do próprio site (ou seja, documentos locais). Para isso, devemos especificar o URL do arquivo desejado a partir da pasta raiz por meio da tag **href**.

Por exemplo, o usuário poderá clicar no termo **Cadastro** para acessar a página desejada, já que esse termo possui propriedades de vínculo definidas pelas tags **<a>** e ****:

```
<a href="interno/cadastro.html" alt="Cadastro">
        Cadastro
</a>
```

6.3.3. Link com imagem

Assim como os textos, as imagens também podem ser definidas como vínculos para um determinado arquivo. Entretanto, em vez de utilizarmos textos, devemos utilizar as tags destinadas à inclusão de imagem para que seja criado o link desejado. Uma imagem pode ser incluída em um documento HTML5 por meio da tag **<img />** e com alguns atributos, como **src** (referente ao caminho da imagem) e **alt** (relacionado à descrição desse arquivo).

No exemplo a seguir, o arquivo **logotipo.jpg** será utilizado como link de acesso ao site da Impacta Certificação e Treinamento. O uso das tags **<a>** e **** determina propriedades de vínculo para essa imagem:

```
<a href="http://www.impacta.com.br" alt="Site da Impacta">
        <img src="images/logotipo.jpg" alt="Logotipo da Impacta"/>
</a>
```

Nos links em forma de texto, é possível distinguir quais vínculos já foram visitados por meio de cores destacadas. Nas imagens, essa diferenciação pela cor também acontece, só que através de suas bordas. Essa característica pode representar um problema quando trabalhamos com imagens, pois uma simples adição de cor pode comprometer sua exibição na tela. Para eliminar essa associação de cores, devemos especificar o atributo **border="0"** na tag **<img />,** ou aplicá-lo via folhas de estilo (CSS).

Muitas páginas Web utilizam uma imagem de tamanho reduzido como link para o acesso à mesma imagem, só que em dimensões maiores. No entanto, esse redimensionamento não modifica realmente o tamanho do arquivo. Isso significa que um arquivo com 50 pixels de largura e 100 Kb continuará com 100 Kb, mesmo quando aumentamos a largura para 200 pixels.

No próximo exemplo, o texto **[+] ampliar** foi utilizado como vínculo para que o arquivo **fotomaior.jpg** seja exibido no browser:

```
<a href="images/fotomaior.jpg" alt="Ampliar foto">
        [+] ampliar
</a>
```

6.3.4. Link para e-mail

Um texto também pode ser utilizado como vínculo para o envio de mensagens para um endereço de e-mail. Para que isso seja possível, devemos incluir o atributo **mailto:** antes de digitar o endereço destino da mensagem.

Como podemos observar no exemplo a seguir, a expressão **Fale-Conosco** aparece entre as tags **<a>** e **** para que seja possível utilizá-la como vínculo. Dessa forma, assim que o usuário clicar nesse texto, será aberto automaticamente o programa de correio eletrônico padrão com o destinatário, como **seu@email.com.br**:

```
<a href="mailto:seu@email.com.br" alt="Fale-Conosco">Fale-Conosco</a>
```

- **Enviando para mais de um destinatário**

Para que uma mensagem seja enviada para vários destinatários, devemos incluir os endereços desejados, como mostra o exemplo a seguir:

```
<a href="mailto:seu@email.com.br;fulano@fulano.com.br" alt="Fale-Conosco">
    Fale-Conosco
</a>
```

- **Definindo o assunto da mensagem**

Além de especificar o destinatário da mensagem, também podemos indicar o assunto por meio do atributo **subject**, que deve ser inserido após o endereço de e-mail, seguido por um ponto de interrogação:

```
<a href="mailto:seu@email.com.br?subject=HTML5 Fundamentos" alt="Fale-Conosco">
    Fale-Conosco
</a>
```

No exemplo anterior, o assunto enviado para **seu@email.com.br** é **HTML5 Fundamentos**.

- **Enviando mensagens com cópia**

Para que a mensagem eletrônica seja enviada como cópia para outro destinatário, devemos incluir o atributo **cc** da seguinte maneira:

```
<a href="mailto:seu@email.com.br?cc=impacta@impacta.com.br" alt="com cópia">
    Enviando com cópia
</a>
```

E com o atributo **bcc** enviamos com cópia oculta:

```
<a href="mailto:um@impacta.com.br?cc=dois@impacta.com.br&bcc=tres@impacta.com.br" alt="BCC">
    Enviando com cópia oculta
</a>
```

- **Escrevendo no corpo do e-mail**

O vínculo para um endereço de e-mail também permite a inclusão de texto no corpo da mensagem que será enviada. No exemplo a seguir, o texto **Gostaria de mais informações** será colocado no corpo da mensagem por meio do atributo:

```
<a href="mailto:aviseme@impacta.com.br?body=Gostaria de mais informações" alt="">
    Avise-me quando estiver disponível
</a>
```

Quando for necessário agregar mais de um atributo, deve ser utilizado o caractere **&** para concaterná-los.

6.3.5. Nomeando âncoras

Para que seja possível acessar rapidamente um determinado ponto em um documento HTML5, devemos criar um vínculo com a ajuda das tags **<a>** e **** e do atributo **id**:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Nomeando âncoras</title>
        <style type="text/css" rel="Stylesheet">
            section{height:500px;}
        </style>
    </head>
    <body>
        <main role="main">
            <section id="inicio">
                Texto 1
            </section>
            <section id="meio">
                Texto 2
                <a href="cap6_3.html#fim">Texto 3</a>
            </section>
            <section id="fim">
                Texto 3

                <a href="cap6_3.html#inicio">Texto 1</a>
            </section>

        </main>
    </body>
</html>
```

No exemplo anterior, o **id** da **section** serve como referência para os links **Texto 1**, **Texto 2** e **Texto 3**.

O vínculo foi criado a partir da definição do caminho do arquivo, além da utilização do símbolo # seguido pelo nome da âncora.

6.4.Determinando a janela de destino

De acordo com o padrão, o conteúdo do arquivo associado ao link substituirá a janela em execução. Por meio do atributo **target**, é possível determinar outro destino para o vínculo que será pressionado. Esse atributo deve ser descrito da seguinte maneira, entre as tags **<a>** e ****:

```
<a href="cap6_3.html" target="_blank" alt="Nova aba">
    Abrindo em uma nova aba
</a>
```

Neste caso, o atributo **target** pode possuir os seguintes valores: **_blank**, **_self**, **_parent**, **_top**.

6.5.Disponibilizando arquivos para download

O usuário pode, ainda, realizar o download de um arquivo por meio de um vínculo. Para que o browser não seja capaz de reconhecer e exibir o arquivo em tela, este deve ser compactado. Dessa forma, quando o usuário clicar sobre o vínculo, a caixa de download será exibida automaticamente.

No exemplo a seguir, é preciso apenas clicar no texto **DOWNLOAD** para que o download do **arquivo.zip** seja iniciado:

```
<a href="arquivo.zip">DOWNLOAD</a>
```

6.6.Microdata

Com relação à criação de vínculos, um recurso muito importante que foi adicionado à especificação do HTML5 são os microdados (microdata). Este novo recurso permite dar sentido a um conteúdo de texto, enriquecendo a semântica do documento, por criar referências ou vínculos com o texto e seu real significado.

Veja um exemplo de um documento sem o uso de microdata:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Sem utilizar Microdata</title>
    </head>
    <body>
        <main role="main">
            <div>
              Glaucio Daniel
              <img src="images/glaucio.jpg"/>

              Desenvolvedor Web
              <div>
                Av. Paulista, 1009
                São Paulo SP 01311-100
              </div>
              (11) 3254-2200</span>
              <a href="mailto:glaucio@html5dev.com.br">
                glaucio@html5dev.com.br</a>

              Site da Empresa:<a href="http://www.impacta.edu.br">www.impacta.edu.br</a>
            </div>
        </main>
    </body>
</html>
```

Percebemos que as informações do cliente, não possuem nenhum significado para os motores de busca e outros mecanismos Web, são apenas informações que apenas humanos conseguem compreender.

Agora, com o uso de microdata, podemos orientar motores de busca, mecanismos de renderização dos navegadores, explicando o sentido do texto, por meio de dos seguintes atributos:

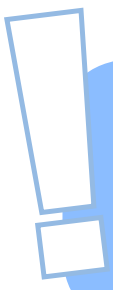
- **itemscope**;
- **itemtype**;
- **itemprop**.

6.6.1. Itemscope

Este atributo é responsável por determinar o início de um escopo. Quando um texto diz respeito a uma pessoa, todas as informações desta pessoa ficam dentro de um elemento **section** ou **div**, que possui o atributo **itemscope**, dando o sentido de que pertencem ao mesmo escopo.

6.6.2. Itemtype

Para determinar o tipo de informação, a coleção microdata possui o atributo **itemtype**, que deve definir o tipo do conteúdo subsequente. O atributo **itemtype** aponta para o site **schema.org** direcionando a especificação do assunto em questão:



Schema.org criou uma coleção de termos utilizados por desenvolvedores Web que podem ser utilizados para dar sentido ao seu documento, tornando-o, assim, amigável aos principais mecanismos de buscas, como Google, Microsoft, Yandex e Yahoo!

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Microdata</title>
    </head>
    <body>
        <main role="main">
            <div itemscope itemtype="http://schema.org/Person">
              <span itemprop="name">Glaucio Daniel</span>
              <img src="images/glaucio.jpg" itemprop="image" />

              <span itemprop="jobTitle">Desenvolvedor Web</span>
              <div itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
                <span itemprop="streetAddress">
                  Av. Paulista, 1009
                </span>
                <span itemprop="addressLocality">São Paulo</span>,
                <span itemprop="addressRegion">SP</span>
                <span itemprop="postalCode">01311-100</span>
              </div>
              <span itemprop="telephone">(11) 3254-2200</span>
              <a href="mailto:glaucio@html5dev.com.br" itemprop="email">
                glaucio@html5dev.com.br</a>

                Site da Empresa:
                <a href="http://www.impacta.edu.br" itemprop="url">www.impacta.edu.br</a>
            </div>
        </main>
    </body>
</html>
```

No exemplo anterior, o elemento **div**, após o elemento **main**, possui dois atributos da coleção microdata: **itemscope** e **itemtype**. O atributo **itemtype** aponta para o endereço da especificação **schema.org/Person**, identificando que aquele bloco de informações diz respeito a uma pessoa.

6.6.3. Itemprop

Este é o atributo responsável por determinar dentro daquele escopo que tipo de informação é o item em questão. No exemplo anterior, **Glaucio Daniel** é um **itemprop=”name”** no escopo de **itemtype=http://schema.org/Person**, significando que **Glaucio Daniel** é o nome de uma pessoa. Embora esta seja uma informação óbvia do ponto de vista humano, para a máquina, apenas com o uso dos microdados é que as informações passam a ter sentido.

Além do benefício semântico para motores de busca, permite que os mesmos integrem tais informações com redes sociais, mapas e lugares (places).

Com os microdados, é possível criar **Rich Snippets**, que são aquelas descrições que aparecem em sites de busca.

Para maiores detalhes sobre as possibilidades de microdados, consulte:

Tipo de Informação	Itemtype e detalhes
Empresas	http://schema.org/Organization
Pessoas	http://schema.org/Person
Produtos	http://schema.org/Product
Eventos	http://schema.org/Event
Resenha (Avaliação)	http://schema.org/Review

Teste seus conhecimentos

Trabalhando com vínculos (links) e microdata

6

1. Qual elemento define o vínculo que permite acesso a um determinado arquivo?

- ☐ a) link
- ☐ b) reference
- ☐ c) itemprop
- ☐ d) a
- ☐ e) microdata

2. Qual o código para criarmos uma referência para um elemento header com o id=”topo” num arquivo chamado artigos.html?

- ☐ a) topo
- ☐ b) topo
- ☐ c) topo
- ☐ d) topo
- ☐ e) topo

3. Qual é a coleção responsável por dar sentido semântico a um texto, criando vínculos com objetos externos localizados em schema.org?

- ☐ a) itemprop
- ☐ b) itemtype
- ☐ c) Microdata
- ☐ d) itemscope
- ☐ e) itemgroup

4. Qual é o atributo responsável por determinar o tipo de uma informação que um escopo irá informar?

- ☐ a) itemtype
- ☐ b) itemprop
- ☐ c) itemscope
- ☐ d) itemgroup
- ☐ e) itemnav.

Mãos à obra!

6 Trabalhando com vínculos (links) e microdata

Laboratório 1

A – Utilizando a coleção microdata

1. Utilizando a coleção microdata, transforme o texto a seguir em uma informação semanticamente legível, utilizando o **itemtype http://www.schema.org/Organization**:

Empresa: Impacta Certificação e Treinamento
Endereço: Av. Paulista, 1009
Tel: (11) 3254-2200
Site: www.impacta.com.br

Trabalhando com listas e tabelas

7

- ✓ Listas;
- ✓ Tabelas.

7.1.Listas

Quando trabalhamos com documentos HTML5, podemos utilizar listas com a finalidade de dividir os elementos de um documento em itens ou categorias. Assim, podemos ter vários tipos de listas ordenando itens da forma desejada. A seguir, veremos alguns tipos de listas com as quais podemos trabalhar.

7.1.1.Lista ordenada

Este tipo de lista permite organizar seus itens, definindo uma ordem para os mesmos de forma explícita. A tag **<ol>** é utilizada para criar uma lista ordenada. Observe a seguir:

```
<ol>
 <li>conteúdo da lista</li>
</ol>
```

ol significa Ordened List.

Os atributos que podem ser utilizados em conjunto à tag **<ol>** são **start** e **type**, cujas descrições são as seguintes:

- **start**: Trata-se de um atributo que permite determinar o valor inicial da numeração de itens da lista. Dessa forma, seu valor deve ser o primeiro número a partir do qual será iniciada a ordenação;

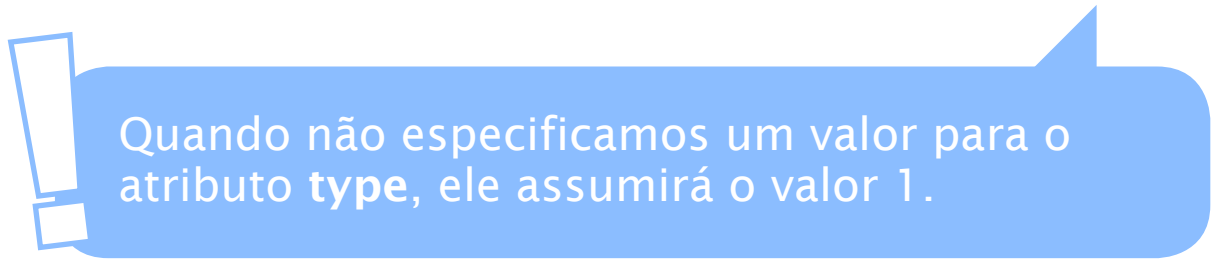
- **type**: Trata-se de um atributo que permite determinar o tipo de lista a ser utilizado. Os valores que podem ser utilizados com este atributo são os seguintes:

  - **A**: Determina a utilização das letras do alfabeto em caixa alta;

  - **a**: Determina a utilização das letras do alfabeto em caixa baixa;

  - **I**: Determina a utilização dos algarismos romanos em caixa alta;

- **i**: Determina a utilização dos algarismos romanos em caixa baixa;
- **1**: Determina a utilização de números para ordenar os itens da lista.




```
<ol type="a">
 <li> primeiro</li>
 <li> segundo</li>
 <li> terceiro</li>
</ol>
```

O resultado será o seguinte:

a. primeiro
b. segundo
c. terceiro

Este exemplo determina a ordenação da lista por meio de letras em caixa alta, sendo que esta ordenação inicia-se a partir da terceira letra do alfabeto, conforme podemos observar na figura a seguir:

```
<ol type="A" start="3">
 <li> primeiro</li>
 <li> segundo</li>
 <li> terceiro</li>
</ol>
```

O seguinte resultado é apresentado no navegador:

C. primeiro
D. segundo
E. terceiro

7.1.2. Lista não ordenada

Além das listas ordenadas, contamos com as listas não ordenadas. Com elas, não podemos definir uma ordem de forma explícita.

A tag utilizada para criar uma lista não ordenada é a **<ul>**, conforme podemos observar a seguir:

```
<ul>
 <li>conteúdo da lista</li>
</ul>
```

Irá produzir o seguinte resultado:

- conteúdo da lista

Já o exemplo a seguir demonstra como definir o marcador da lista como sendo um círculo vazado:

```
<ul type="circle">
 <li> primeiro</li>
 <li> segundo</li>
 <li> terceiro</li>
</ul>
```

Ao executarmos este exemplo, o resultado obtido é o seguinte:

- primeiro
- segundo
- terceiro

7.1.3. Lista de definição

Este tipo de lista permite não apenas a exibição de itens, mas também de subitens, sendo bastante utilizada quando trabalhamos com títulos e subtítulos.

Observe o exemplo a seguir, em que trabalhamos com a exibição de um índice contendo itens e subitens:

```
<dl>
 <dt> Capitulo 1 </dt>
    <dd> Primeiro item</dd>
    <dd> Segundo item</dd>
 <dt> Capitulo 2 </dt>
    <dd> Primeiro item</dd>
    <dd> Segundo item</dd>
</dl>
```

Em que:

- DL: Definition List;
- DT: Definition Term;
- DD: Definition Description.

Com a execução deste exemplo, o seguinte resultado é apresentado a partir do navegador:

Capitulo 1
    Primeiro item
    Segundo item
Capitulo 2
    Primeiro item
    Segundo item

7.2.Tabelas

Para criarmos uma tabela em um documento HTML5, utilizamos o elemento **<table>**. Trabalhar com tabelas é uma tarefa que permite exibir informações de forma tabulada e criar uma listagem seguida de alguns dados a seu respeito.

A estrutura básica de uma tabela é a seguinte:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Tabelas</title>
        <style type="text/css" rel="Stylesheet">
            table{width: 99%;}td{text-align: center;}
        </style>
    </head>
    <body>
        <table>
            <thead>
                <th>
                    Código
                </th>
                <th>
                    Nome
                </th>
            </thead>
            <tbody>
                <tr>
                    <td>
                        1
                    </td>
                    <td>
                        Marcos Lima
                    </td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

Em que:

- **<table>**: Refere-se à tag que permite criar uma tabela;
- **<thead>**: Refere-se ao cabeçalho da tabela;
- **<th>**: Refere-se ao cabeçalho de uma coluna;
- **<tbody>**: Refere-se ao corpo da tabela;
- **<tr>**: Refere-se à tag que permite criar uma linha para a tabela, na qual podemos inserir as colunas desejadas. **tr** significa Table Row;
- **<td>**: Refere-se à tag que permite criar uma coluna na linha da tabela. Portanto, ela representa uma célula da tabela, na qual são inseridas as informações desejadas. **td** significa Table Data.

7.2.1. Formatação de uma tabela

Com o advento da CSS e sua versão CSS3, toda a parte de estilização de uma tabela, que antes era feita direto no elemento, passa a ser realizada via folhas de estilos (CSS).

7.2.2. Mesclando células

Podemos combinar as células de uma tabela entre si a fim de formar uma célula única. Este processo é denominado mescla de células. Mais detalhes a esse respeito serão abordados a seguir.

7.2.2.1. Mesclando colunas

O processo que permite mesclar colunas requer a definição da quantidade de colunas que uma determinada célula pode ocupar, sem que o layout da tabela sofra quaisquer alterações.

Desta forma, caso determinemos que uma célula deve ocupar o espaço de três colunas, as próximas duas colunas devem deixar de ser utilizadas, pois a célula mesclada ocupará o lugar das três colunas.

```
<td colspan="x">informação</td>
```

Em que **x** indica o número de colunas que desejamos mesclar. Observe a mescla no exemplo a seguir:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Tabelas</title>
        <style type="text/css" rel="Stylesheet">
            table{width: 99%; border:solid 2px;}td{text-align: center;
            width: 50%;border:solid 1px;}
            th{text-align: center; border:solid 1px;}
        </style>
    </head>
    <body>
        <table>
            <thead>
                <th colspan="2">
                    Relatório do Cliente
                </th>
            </thead>
            <tbody>
                <tr>
                    <td>
                        1
                    </td>
                    <td>
                        Marcos Lima
                    </td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

O resultado será semelhante ao produzido abaixo:

Relatório do Cliente	
1	Marcos Lima

7.2.2.2. Mesclando linhas

Assim como a mescla de colunas, a mescla de linhas é um processo que requer a determinação de quantas linhas devem ser ocupadas pela nova célula mesclada, sem que o layout da tabela sofra quaisquer alterações.

Observe o exemplo a seguir:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Tabelas</title>
        <style type="text/css" rel="Stylesheet">
            table{width: 99%; border:solid 2px;}td{text-align: center;
             width: 25%;border:solid 1px;}
            th{text-align: center; border:solid 1px;}
        </style>
    </head>
    <body>
        <table>
            <tbody>
                <tr>
                    <td rowspan="3">Antenor</td>
                </tr>
                <tr>
                    <td>Jan</td>
                    <td>
                        R$ 50000.00
                    </td>
                </tr>
                <tr>
                    <td>Fev</td>
                    <td>
                        R$ 28430.00
                    </td>
                </tr>
            </tbody>
        </table>
    </body>
</html>
```

Antenor	Jan	R$ 50000.00
	Fev	R$ 28430.00

Teste seus conhecimentos

Trabalhando com listas e tabelas

7

1. Qual elemento utilizamos para criar uma lista ordenada por letras em caixa baixa?

- ☐ a) <ol type=”A”>
- ☐ b) <ol type=”1”>
- ☐ c) <ol type=”A” start=”5”>
- ☐ d)
- ☐ e) <ol type=”a”>

2. Qual elemento e atributo utilizamos para criar uma lista não ordenada com um marcador representado por um círculo vazado?

- ☐ a) <ul type=”circle”>
- ☐ b) <li type=””>
- ☐ c) <ul type=”circulo”>
- ☐ d) <dl id=”circle”>
- ☐ e)

3. Qual elemento utilizamos com o elemento <table> para delimitar linhas de uma tabela?

- ☐ a) <thread>
- ☐ b) <tbody>
- ☐ c) <th>
- ☐ d) <tr>
- ☐ e) <td>

4. Qual atributo utilizamos para mesclar colunas de uma tabela?

- ☐ a) border = solid
- ☐ b) border = duo
- ☐ c) colspan
- ☐ d) rowspan
- ☐ e) rowspan e colspan

Mãos à obra!

Trabalhando com listas e tabelas

7

Laboratório 1

A – Criando uma tabela em um documento HTML5

1. Utilizando os conhecimentos adquiridos sobre tabelas, crie um documento HTML5 com uma tabela, conforme a imagem abaixo:

Relatório Financeiro				
Cliente	**Ano Apuração**	**Total Vendido**	**Total Pago**	**VIP**
HTML5 Dev	2013	R$ 500.000,00	R$ 459.000,00	sim

Introdução ao CSS3 8

- ✓ Conceito de camadas
- ✓ Folhas de estilo (CSS);
- ✓ Estilos CSS;
- ✓ Cascateamento;
- ✓ Inserção de comentários;
- ✓ Atributos para formatação;
- ✓ Seletores;

8.1.Introdução

Sites da Web, sejam eles redes sociais, comércio eletrônico, portais de notícias ou outros, produzem milhões de novas informações todos os dias. Além de gerar conteúdo novo, tais sistemas são obrigados a atualizar periodicamente as páginas já existentes. Tamanha responsabilidade implica no envolvimento de vários profissionais, incluindo diagramadores, redatores, programadores e editores.

Uma mudança gráfica no layout do site, por exemplo, exigiria muito trabalho e tempo. A alteração manual dos arquivos de um site não é apenas uma tarefa trabalhosa, mas também suscetível a erros, principalmente quando lidamos com sites ou Web apps, cujos usuários produzem informação.

Para amenizar a complexidade dessas modificações, organizar o layout e deixá-lo limpo e atual, muitos profissionais utilizam as folhas de estilo. Estas podem ser bastante úteis quando precisamos alterar elementos do mesmo tipo, mas que pertencem a documentos diferentes.

Nesta leitura, abordaremos como trabalhar com folhas de estilo em cascata (CSS – Cascading Style Sheets) e com seus recursos principais.

8.2.Conceito de camadas

Segundo a definição desse conceito, os elementos de programação com a mesma finalidade devem ser agrupados no mesmo local. Além de facilitar a inclusão de modificações, o conceito de camadas determina uma estrutura mais organizada para o sistema. Dessa forma, é possível acrescentar ou atualizar informações sem comprometer o conteúdo atual.

A partir do conceito de camadas, é possível trabalhar com as folhas de estilo em cascata. As folhas de estilo CSS definem como o sistema será formatado de modo a facilitar a visualização do código e a organização do sistema como um todo.

8.3.Folhas de estilo (CSS)

A folha de estilos em cascata CSS é um documento responsável pela definição de critérios de formatação de páginas. Com isso, é possível estabelecer um padrão de formatação que será associado às páginas desejadas. A extensão utilizada para os arquivos de folhas de estilo é a **.css**.

Por meio de um arquivo de folha de estilo, podemos definir a formatação dos elementos referente à cor, fonte, tamanho, posicionamento e outros estilos de um documento HTML. No entanto, para que tais padrões sejam aplicados, será necessário criar um vínculo entre a folha de estilo e as páginas que serão formatadas.

8.3.1. Declarando estilos

Para que uma tag seja formatada por meio de uma folha de estilo, será preciso declarar a sintaxe a seguir:

```
elemento {
    atributo:valor;
    atributo:valor2;
}
```

Em que:

- **elemento:** Trata-se da descrição do elemento que será formatado. Neste caso, este elemento será uma tag HTML sem os sinais de menor e maior;

- **atributo:** Define qual será o atributo utilizado para a aplicação do estilo, desde que o nome seja de um atributo CSS válido, como o tamanho da fonte (font-size);

- **valor:** Refere-se ao valor específico do atributo definido, como 15 pt (ou seja, 15 pontos) para o atributo de tamanho de fonte (font-size).

8.4.Estilos CSS

Os estilos de uma CSS podem ser aplicados em uma linha ou cabeçalho da própria página HTML ou por meio de um arquivo externo, o que é mais indicado.

8.4.1.Declarando estilos internamente

As folhas de estilo podem ser definidas dentro do próprio documento que será formatado. Isso significa que os estilos associados a uma tag serão sempre aplicados de acordo com os mesmos parâmetros ao longo do arquivo. É importante lembrar que essas definições internas são prioritárias em comparação às externas, ou seja, elas serão utilizadas primeiramente, caso também existam vínculos para uma CSS externa.

8.4.2.Declarando estilos no cabeçalho

As definições das tags podem ser descritas no cabeçalho do documento que sofrerá as alterações. Todas as tags seguintes com o mesmo nome serão exibidas da mesma maneira, cada qual com os seus respectivos parâmetros. A descrição dos parâmetros também é feita sempre da mesma forma para todos os modos de vínculo, como podemos observar no exemplo a seguir:

```
<head>
    <style type="text/css" rel="StyleSheet">
        p {
          font-size : 20pt;
          font-family : “verdana”;
          color : #FF0000;
          text-align : center;
        }
    </style>
</head>
```

De acordo com o exemplo anterior, a tag **<p>** terá fonte Verdana com tamanho de 20 pt, cor vermelha e alinhamento central.

8.4.3. Declarando estilos diretamente na tag

Uma folha de estilo é classificada como inline quando as regras são declaradas dentro das próprias tags de um elemento HTML. Esse método deve ser utilizado apenas para aplicar um estilo a um só elemento, pois o usuário poderá confundir facilmente o conteúdo da tag com as regras de formatação, já que ambos estão no mesmo local.

É importante mencionar que essa não é uma boa prática de desenvolvimento Web, uma vez que se o estilo for aplicado diretamente no elemento, o código fica engessado, logo este recurso deve ser utilizado com cautela e em casos de exceções.

A sintaxe utilizada para a aplicação da folha de estilo inline pode ser vista a seguir:

```
<elemento style="atributo: valor"> </elemento>
```

No seguinte exemplo, será formatado um determinado parágrafo de modo que sua cor fique vermelha e a margem tenha 3 px (pontos):

```
<p style="color:#FF0000; margin: 3px;">texto </p>
```

8.4.4. Declarando estilos externamente

A definição de uma folha de estilo por meio de um arquivo externo é a maneira mais recomendada para trabalhar com esse tipo de recurso. Para que seja possível o uso de arquivos externos, devemos criar vínculos com os documentos que serão formatados. Com isso, todos os documentos vinculados à folha de estilo serão modificados de acordo com os valores configurados nas tags.

Um arquivo de folha de estilo com a extensão **.css** pode ser criado em qualquer programa editor de textos. Sua estrutura pode ser vista a seguir:

```
elemento {
    atributo1: valor;
}
```

O próximo exemplo facilita a compreensão dessa estrutura:

```
body {
    background-color: #FF0000;
 }

p {
    color: #FFFFFF;
    font-size: 20pt;
    text-align: center;
 }
```

De acordo com o exemplo apresentado, o parágrafo terá a fonte com tamanho de 20 pt, cor branca e alinhamento centralizado. O fundo da página, por sua vez, será vermelho.

Para vincular a folha de estilo ao documento HTML, devemos utilizar a tag **<link />**. Os parâmetros dessa tag incluem o caminho de localização do arquivo **.css** e a descrição do tipo de arquivo externo, como podemos observar no próximo exemplo:

```
<head>
 <link type="text/css" href="arquivo.css" rel="stylesheet" />
</head>
```

Como é possível verificar no exemplo anterior, a tag **<link />** deve ser inserida no cabeçalho (**head**) do documento HTML. Graças à correta localização e descrição da folha de estilo, os padrões CSS poderão ser importados para aplicação nesse documento.

8.5.Cascateamento

Um documento HTML5 pode ser vinculado a uma ou várias folhas de estilo. Caso mais de uma CSS esteja associada a esse documento, o browser será obrigado a decidir quais são os padrões mais importantes, uma vez que a mesma regra pode aparecer em várias folhas de estilo.

A prioridade entre definições de estilo atende às seguintes regras, descritas em ordem de importância:

- Os estilos aplicados caso não existam outros padrões que possam ser aplicados no seu lugar: são os estilos definidos por omissão;

- Os estilos definidos por meio de uma folha de estilo interna (ou seja, dentro do elemento **<style>**) ou por meio de um arquivo externo;

- Os estilos definidos nos elementos de um documento HTML por meio do atributo **<style>** em uma folha de estilo inline.

Como podemos notar, os estilos definidos através do atributo **<style>** no próprio elemento são prioritários em relação aos demais estilos. Caso o browser reconheça diferentes versões para o mesmo estilo, será utilizada somente a mais recente.

A ordem de maior utilização dos estilos pode ser categorizada de maneira simples:

- **Externo:** Arquivo **.css**;

- **Incorporado:** Através da tag **<style>** declarada no cabeçalho da página;

- **Inline:** Tag **<style>** declarada em uma determinada linha do código da página.

8.6.Inserindo comentários

Para facilitar a compreensão de um determinado código, é possível incluir comentários nas definições de estilo CSS. Tais comentários explicam o código que será escrito, de forma que outros usuários possam entender rapidamente sua utilização.

Um comentário deve ser incluído da seguinte forma: em primeiro lugar, devemos escrever a sequência de caracteres /*. Em seguida, é preciso incluir a descrição do comentário. Depois de incluir a descrição do comentário, devemos fechá-lo escrevendo a sequência de caracteres */, como mostra o exemplo a seguir:

```
<style type="text/css">
p {
 /* isso é um comentário*/
 text-align: center
}
 </style>
```

É importante utilizar comentários para documentação do código, entretanto em versões minificadas (utilizadas em ambiente de produção), os comentários e espaços são retirados.

8.7.Atributos utilizados para formatação

Os arquivos de folhas de estilo podem apresentar atributos associados a diferentes estilos de formatação, como aprenderemos a partir de agora.

8.7.1.Declaração de cores

Uma folha de estilo permite que sejam utilizados diferentes métodos para a especificação de cores:

- **Nome da cor:** Uma cor pode ser especificada por meio do seu nome, como **green**;
- **#valor:** Determina um valor padronizado da cor que será utilizada;
- **Rgb(vermelho, verde, azul):** Define os valores de vermelho, verde e azul para a composição da cor final, como **rgb (0,255,0)**;
- **rgb(vermelho%,verde%,azul%):** Define os valores de vermelho, verde e azul como porcentagens em relação ao valor máximo de cada cor. Por exemplo, **rgb(15%,40%,30%)**;
- **Nome resumido:** Caso seja uma cor segura, pode ser declarado em vez de #**CC6699** como #**C69**;
- **rgba(vermelho, verde, azul, alpha):** Define os valores de vermelho, verde e azul semelhante ao **rgb**, porém o último parâmetro **alpha**, determina a transparência da cor, em que **1.0** é 100% visível, **0.50** é 50% transparente e **0.20** é 80% transparente.

8.7.2. Formatando texto

Vários atributos podem ser utilizados para a formatação do texto de um elemento. Veja quais são esses atributos:

Atributos	Valor	Definição
color	Valor em formato hexadecimal o nome da cor.	Cor do texto.
font-family	Nome da fonte.	Tipo da fonte.
font-size	Valor referente ao tamanho da fonte ou porcentagem.	Tamanho da fonte.
font-style	**normal**, **oblique** ou **italic**.	Estilo do texto.
font-variant	**normal** ou **small-caps**.	Os caracteres são convertidos para maiúsculo.
font-weight	**normal**, **bold**, **bolder**, **lighter**, **[100 -900]**.	Estilo negrito.
letter-spacing	**normal** ou **pt**.	Valor para o espaço disponível entre as letras.
text-align	**left**, **right**, **center** ou **justify**.	Ajuste do alinhamento do texto.
text-decoration	**overline**, **underline**, **line-through** ou **blink**.	Decoração do texto ou hiperlink.
text-ident	Porcentagem ou valor desejado para o comprimento.	Identação do texto.
text-transform	**capitaliza**, **lowercase** ou **uppercase**.	Conversão do texto em formato minúsculo ou maiúsculo ou apenas a primeira letra.
vertical-align	**bottom**, **middle**, **sub**, **super**, **text-bottom**, **text-top** e **top**	Alinhamento vertical do texto.

No exemplo a seguir, o documento **style.css** possui alguns dos atributos que foram descritos anteriormente:

```
<style type="text/css">
p{
    font-family: verdana;
    font-size: 20pt;
    font-variant: small-caps;
    font-style: italic;
    color: #f00;
    text-align: center;
    word-spacing: 20px
}
 </style>
```

Já o arquivo **aula8_1.html** possui a seguinte estrutura:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Imagens</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <header>
            <h1>Trabalhando com CSS</h1>
        </header>
        <main role="main">
            <p>
                Texto formatado por folhas de estilo.
            </p>
        </main>
    </body>
</html>
```

O resultado do código renderizado pode ser visto a seguir:

Trabalhando com CSS

Texto formatado por folhas de estilo.

8.7.3. Formatando o plano de fundo

Para que o plano de fundo de uma página seja formatado com as regras CSS, podemos utilizar vários atributos em conjunto com o comando background:

Atributos	Valor	Definição
background-attachment	**fixed**, **scroll** ou **local**.	Define se a imagem de fundo permanecerá fixa enquanto a barra de rolagem é deslizada (**fixed**) ou se ela acompanhará esse movimento (**scroll**).
background-color	**transparent**, valor em formato hexadecimal, **rgb**, **rgba** ou nome da cor.	Determina a cor de fundo de um elemento.
background-image	**url**, **linear-gradient**	Uma imagem que será utilizada como plano de fundo. Podemos utilizar a funcionalidade **linear-gradient** que permite criar degradê utilizando apenas CSS3.
background-repeat	**repeat-x**, **repeat-y**, **repeat**, **space**, **round**, **no-repeat**.	Especifica como a imagem de fundo deve se comportar caso seja maior que a área do elemento em que foi aplicada.
background-size	Valor em pixels, porcentagem ou auto.	Especifica o tamanho das imagens de fundo.
background-position	**center**, [**left** | **right** | **top** | **bottom**] **[%]**	Caso a propriedade **background-image** seja especificada, podemos determinar a localização da imagem e também o seu comportamento quando a janela for redimensionada.

A seguir veja um exemplo do arquivo **style.css**:

```
.btn {

color:#fff;
border-radius:4px;
border:1px solid #980021;
box-shadow: inset 0 2px 3px 0 rgba(255,255,255,.3),
inset 0 -3px 6px 0 rgba(0,0,0,.2),
0 3px 2px 0 rgba(0,0,0,.2);
cursor: pointer;
}

.btn-large{
    width:180px;
    padding: 10px 10px;
    height:50px;
    font-size:20px;
    text-align:center;
}

.btn-primary{
    background-image:
    linear-gradient(to bottom, #E44D3A, #A7372B 130%);
}
```

E com o código HTML carregando os estilos e criando um botão para aplicar os estilos criados:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Imagens</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <header>
            <h1>Trabalhando com CSS</h1>
        </header>
        <main role="main">
            <p>
                <button class="btn btn-large btn-primary" id="btn-envio">
                    Enviar
                </button>
            </p>
        </main>
    </body>
</html>
```

Após a renderização, o código anterior produzirá o seguinte resultado:

8.7.4. Formatando bordas

As bordas podem ser ajustadas de acordo com uma série de atributos, como podemos observar na tabela a seguir:

Atributos	Valor	Definição
border-bottom	**border-bottom-width**, **border-color** e **border-style**.	Todas as propriedades da linha de limite inferior da borda podem ser escritas em uma só declaração.
border-color	Cor	As cores das quatro linhas de limite são escolhidas.
border-style	**none**, **hidden**, **dotted**, **dashed**, **solid**, **double**, **groove**, **ridge**, **inset** ou **outset**.	O estilo das quatro linhas de limite da borda pode ser ajustado com um a quatro valores.
border-top	**border-color**, **border-style** e **border-top-width**.	Todas as propriedades da linha de limite superior são definidas por meio de uma só declaração.
border-radius	[px | %] [px | %]	Determina o grau da curva das margens de um objeto.

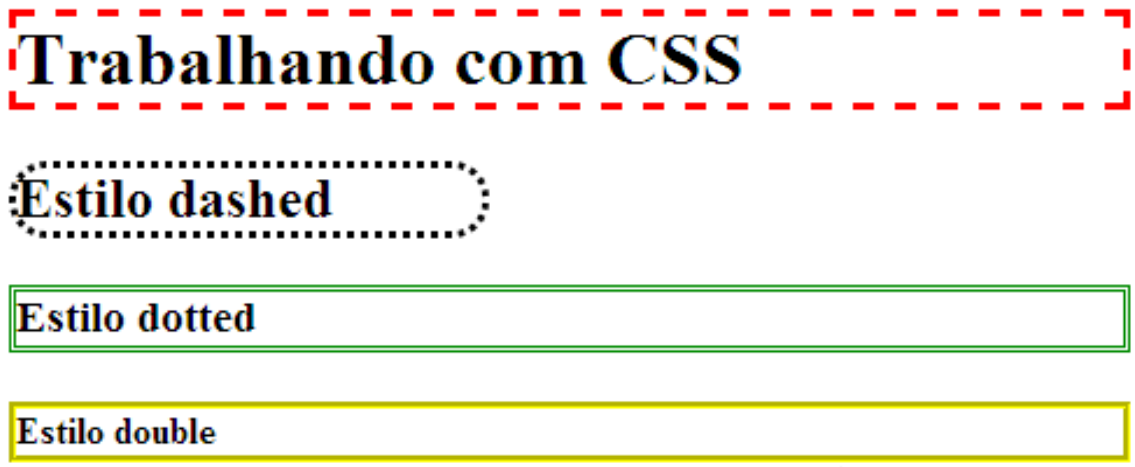
Veja um exemplo com o uso da formatação de bordas:

```css
.borda-arredondada{
    border-radius: 25px;
    width:200px;
}

h1 {
    border-style: dashed;
    border-color: red
 }
 h2 {
    border-style: dotted;
    border-color: black
 }
 h3 {
    border-style: double;
    border-color: green
 }
 h4 {
    border-style: groove;
    border-color: yellow
 }
```

O arquivo **aula8_3.html** com o conteúdo do documento HTML:

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - Imagens</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <header>
            <h1>Trabalhando com CSS</h1>
        </header>
        <main role="main">
            <p>
                <h2 class="borda-arredondada">Estilo dashed</h1>
                <h3> Estilo dotted</h2>
                <h4> Estilo double</h3>
            </p>
        </main>
    </body>
</html>
```

Confira o resultado criado no navegador:



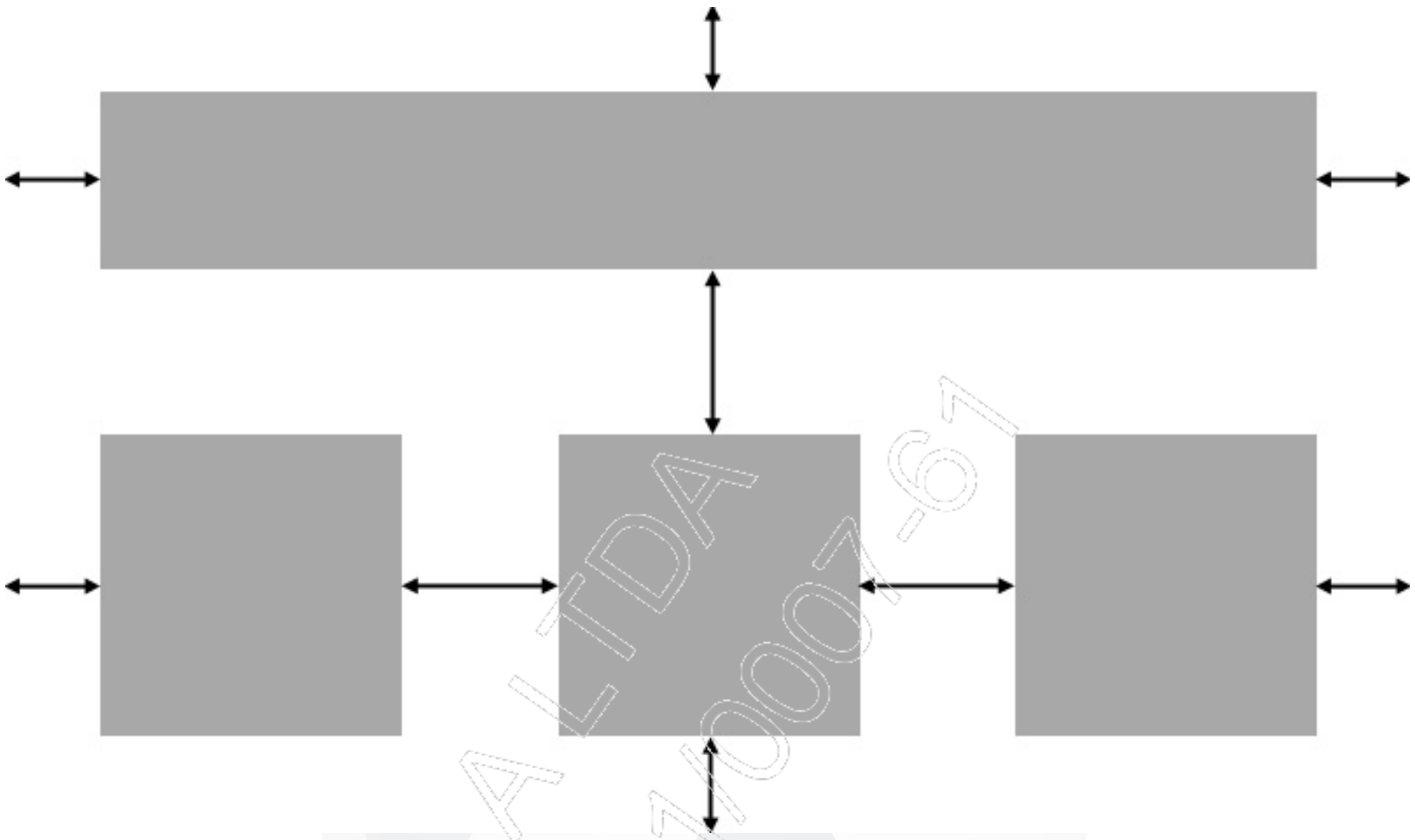
Observe no exemplo acima que o elemento h1 está chamando a classe **borda-arredondada**, criando assim um efeito com cantos arredondados sem o uso de imagens, o que não era possível em versões anteriores ao HTML5.

8.7.5. Formatando espaçamento entre conteúdo, margem e bordas.

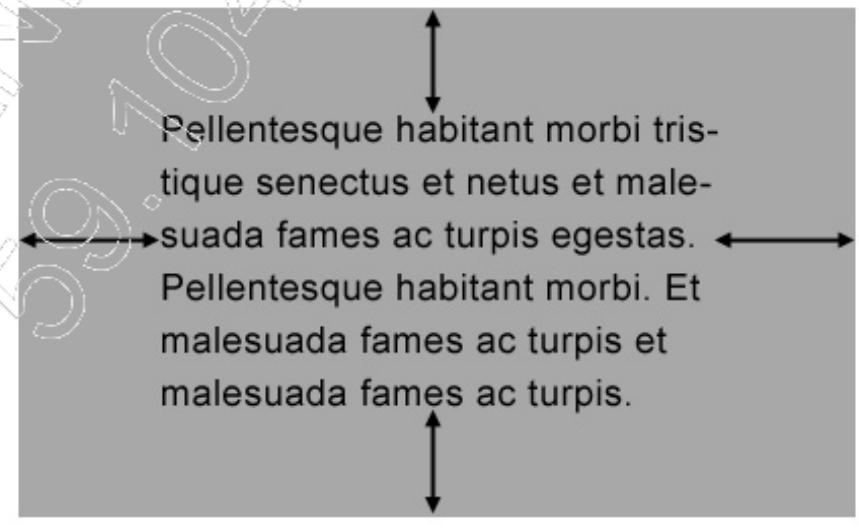
O espaço existente entre as bordas e o conteúdo dos elementos de um documento HTML pode ser ajustado conforme os seguintes atributos:

Atributos	Valor	Definição
padding-bottom	Porcentagem ou valor desejado para o cumprimento.	O espaço inferior entre o conteúdo e as bordas é configurado.
padding-left	Porcentagem ou valor desejado para o cumprimento.	O espaço do lado esquerdo entre o conteúdo e as bordas é configurado.
padding-right	Porcentagem ou valor desejado para o cumprimento.	O espaço do lado direito entre o conteúdo e as bordas é configurado.
padding-top	Porcentagem ou valor desejado para o cumprimento.	O espaço superior entre o conteúdo e as bordas é configurado.

- **Definição das margens de um objeto: margens a partir do topo, da base, da esquerda e da direita**



- **Definição do padding de um bloco**



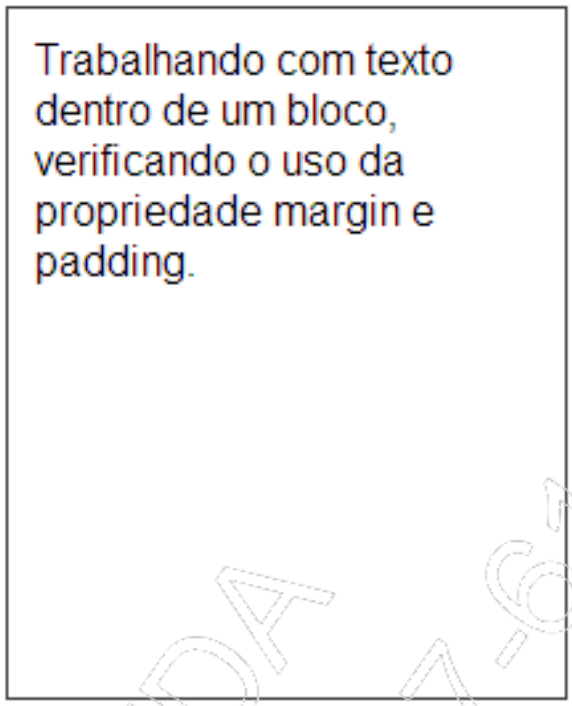
Criamos o estilo em um arquivo **style.css**:

```css
.bloco{
    width:200px;
    height: 250px;
    border:solid 1px #333;
    font-family: arial;
    margin:10px 10px;
    padding-left:10px;
    padding-top: 10px;
}
```

Criamos o conteúdo HTML no arquivo **aula8_4.html**, definindo uma **section** com uma chamada para a classe **bloco**:

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <p>
                <section class="bloco">
                    Trabalhando com texto dentro de
                    um bloco, verificando o uso da propriedade
                    margin e padding.

                </section>
            </p>
        </main>
    </body>
</html>
```

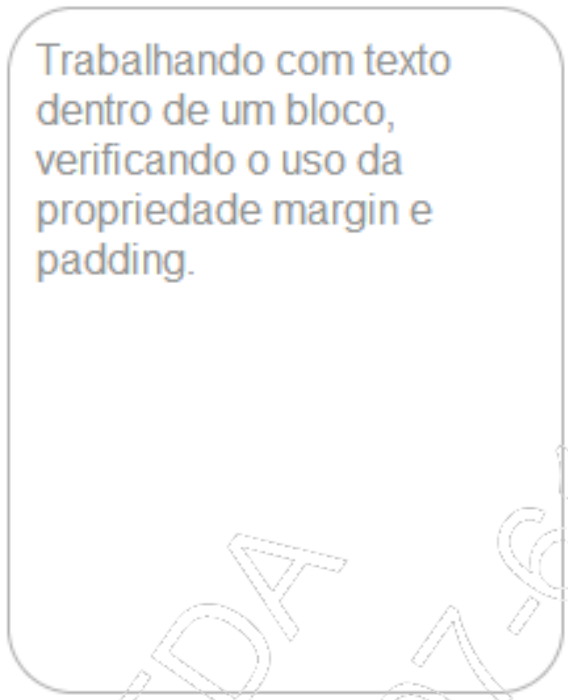
E temos como resultado um bloco definindo as margens externas (**margin**) e internas (**padding**):



Com o clareamento da fonte e a adição de cantos arredondados (**border-radius**), temos um efeito interessante:

```
.bloco{
    width:200px;
    height: 250px;
    border:solid 1px #aaa;
    color:#888;
    font-family: arial;
    margin:10px 10px;
    padding-left:10px;
    padding-top: 10px;
    border-radius: 25px;
}
```

E o resultado renderizado será o seguinte:



As margens de um documento HTML também podem ser formatadas com a ajuda das folhas de estilo. Para isso, será necessário utilizar o atributo **margin**, como podemos observar no exemplo anterior.

8.7.6. Formatando links

Os links inseridos em um documento HTML também podem ser formatados. Para isso, algumas regras de estilo devem ser declaradas para as pseudoclasses do elemento **<a>** do HTML. As pseudoclasses permitem acrescentar efeitos a uma instância dos seletores ou aos próprios seletores.

As pseudoclasses dos links podem ser divididas em:

- **a:hover**: Define a formatação do link no momento em que passamos o mouse sobre ele;
- **a:link**: Define a formatação do link antes dele ser visitado;
- **a:visited**: Define a formatação do link depois que ele é visitado;
- **a:active**: Link ativo.

A sintaxe utilizada para a formatação de links é a seguinte:

```
seletor:pseudo-classe {propriedade: valor}
```

De acordo com o padrão, os links aparecem na cor azul e sublinhados. Para que ambas as características sejam exibidas apenas no momento em que passamos o mouse sobre o link, devemos utilizar o atributo **text-decoration**.

A seguir, podemos visualizar um exemplo desse tipo de formatação, no arquivo **style.css**:

```
a:link {text-decoration: none; color: red}
a:visited {text-decoration: none; color: green}
a:hover {text-decoration: underline; color: blue}
```

A estrutra HTML será utilizada da seguinte maneira:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <p>
             <a href="http://www.impacta.com.br">
              Impacta Certificação e Treinamento
             </a>
            </p>
        </main>
    </body>
</html>
```

Confira o resultado renderizado:

- **Link ainda não visitado**

Impacta Certificação e Treinamento

- **Link já visitado**

  Impacta Certificação e Treinamento

- **Link ao passar o mouse**

  Impacta Certificação e Treinamento

8.8.Seletores

Os seletores permitem não só a definição de diferentes formatações em um mesmo elemento HTML, mas também a criação de estilos personalizados. Ambos os procedimentos serão vistos com detalhes a partir de agora.

8.8.1.Seletores de classe

Por meio dos seletores de classe, é possível configurar diferentes estilos no mesmo elemento de um documento HTML. Por exemplo, suponhamos que vários tipos de alinhamento de parágrafo serão definidos: um ao centro, outro à esquerda e um último à direita. Vejamos como os seletores de classe podem ser aplicados para que essa formatação seja realizada.

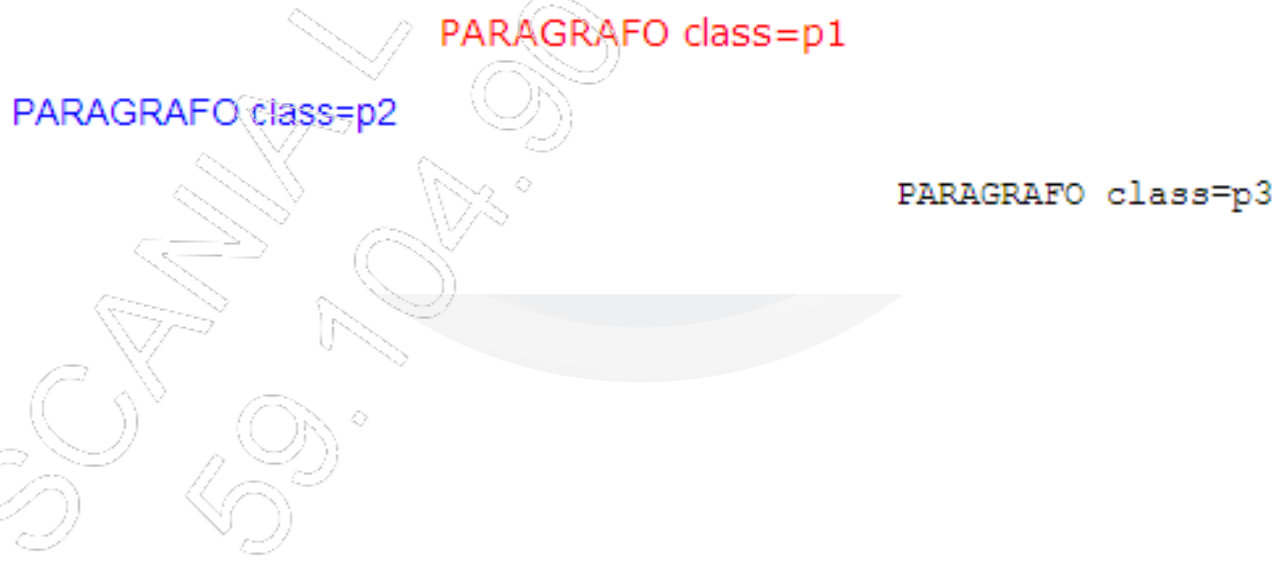
No primeiro exemplo, temos a seguinte estrutura no documento CSS:

```
.p1 {
 font-family: verdana;
 font-size: 14px;
 color: red;
 text-align: center
}
.p2 {
 font-family: arial;
 font-size: 14px;
 color: blue;
 text-align: left
}
.p3 {
 font-family: courier;
 font-size: 14px;
 color: black;
 text-align: right
}
```

O **arquivo.html** apresentará a seguinte estrutura:

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <p class="p1"> PARAGRAFO class=p1 </p>
            <p class="p2"> PARAGRAFO class=p2 </p>
            <p class="p3"> PARAGRAFO class=p3 </p>

        </main>
    </body>
</html>
```

E o resultado renderizado será apresentado da seguinte forma:



8.8.2. Seletores de id

Ao contrário do seletor de classe, o seletor **id** é aplicado a um único elemento, pois os valores do atributo **id** não podem aparecer mais de uma vez em uma página. Isso significa que a quantidade de elementos que possuem **id** em uma página sempre será zero ou um. Conforme a regra do próximo exemplo, somente um elemento **<p>** com o valor **p1** no atributo **id** poderá ser aplicado pelo estilo.

No documento CSS, temos:

```css
#p1 {
 font-family: verdana;
 font-size: 14px;
 color: red;
 text-align: center
}
#p2 {
 font-family: arial;
 font-size: 14px;
 color: blue;
 text-align: left
}
```

Em seguida, a estrutura do arquivo **aula8_7.html** será:

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <p id="p1"> PARAGRAFO ID=p1 </p>
            <p id="p2"> PARAGRAFO ID=p2 </p>

        </main>
    </body>
```

PARAGRAFO ID=p1

PARAGRAFO ID=p2

Teste seus conhecimentos

Introdução ao CSS3

8

1. Qual código deve ser utilizado para criar uma referência a um arquivo chamado estilo.css?

- ☐ a) <link type="text/css" href="css/estilo.css" rel="StyleSheet"></link>
- ☐ b) <script type="text/css" href="css/estilo.css" rel="StyleSheet"></script/>
- ☐ c) <script language="texto/css" href="css/estilo.css" rel="StyleSheet"/>
- ☐ d) <link language="texto/css" href="css/estilo.css" rel="StyleSheet">
- ☐ e) <link type="text/css" href="css/estilo.css" rel="StyleSheet">

2. Qual(is) propriedade(s) utilizamos para formatar a imagem de fundo de um elemento?

- ☐ a) Utilizamos as propriedades background-color e background-size.
- ☐ b) Utilizamos a propriedade backgroud-image.
- ☐ c) Utilizamos a propriedade background-reppeat.
- ☐ d) Utilizamos as propriedades background-image, background-size e background-repeat.
- ☐ e) Utilizamos as propriedades background-color e background-style.

3. Qual propriedade da CSS utilizamos para deixar os cantos das bordas dos elementos arredondadas?

- ☐ a) border-radius
- ☐ b) font-family
- ☐ c) padding-top
- ☐ d) padding-left
- ☐ e) border

4. Qual código deve ser utilizado para determinar 15px de margem externa a partir da esquerda?

- ☐ a) margin-left{15px;}
- ☐ b) margin: 0px 0px 15px;
- ☐ c) margin-left: 15px;
- ☐ d) padding: {15px;}
- ☐ e) padding-left:15px;

Mãos à obra!

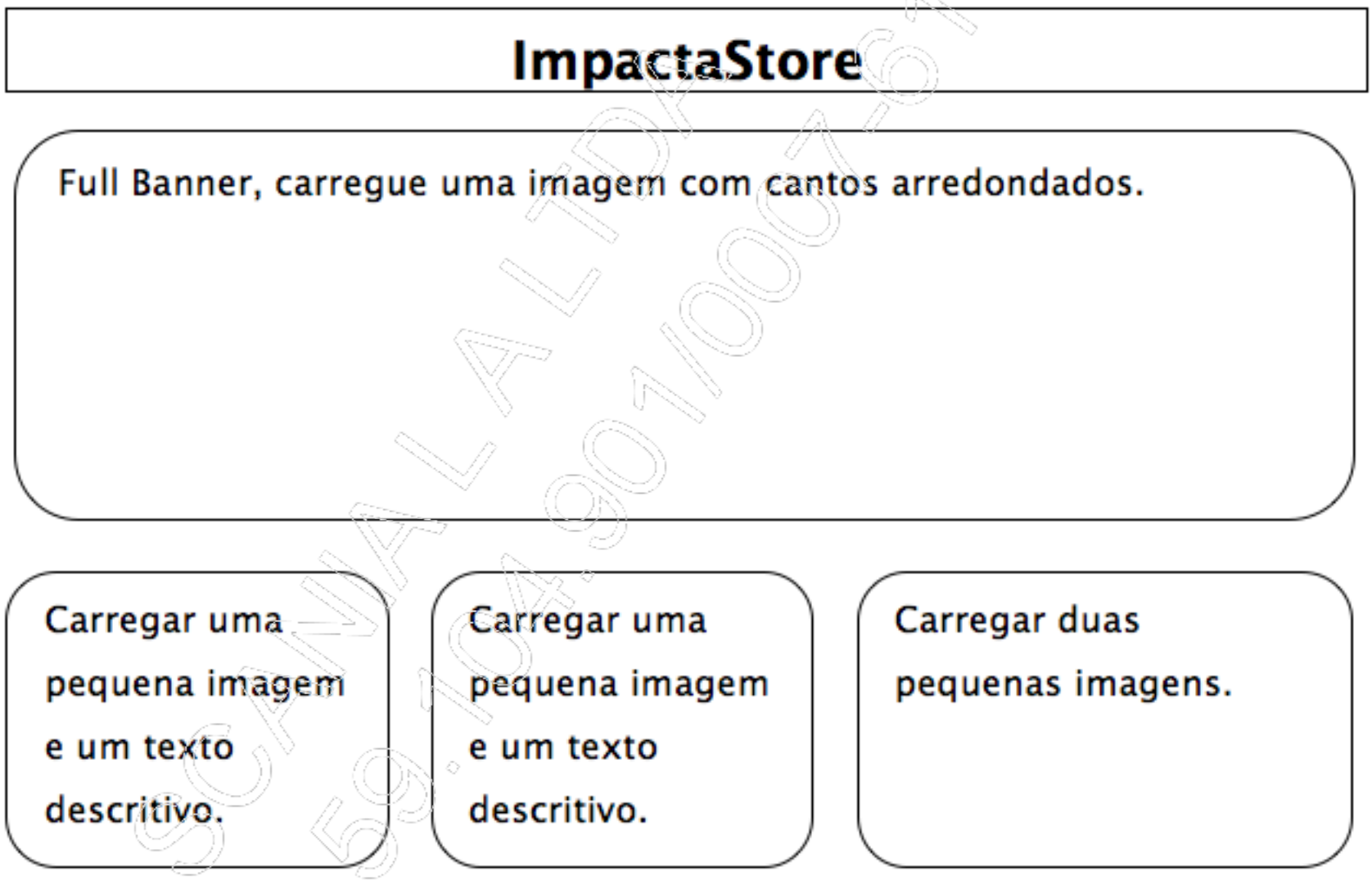
Introdução ao CSS3

8

Laboratório 1

A – Utilizando a CSS

1. Utilizando o wireframe (guia visual de interface) adiante, crie um arquivo chamado **home.html** e um arquivo para folha de estilos denominado **estilo.css**, de forma que produza o seguinte resultado:



Criando e posicionando um layout

9

- ✓ Dimensão real dos elementos;
- ✓ Configuração das margens dos elementos;
- ✓ Posicionamento dos elementos;
- ✓ Definição das camadas;
- ✓ Definição do modo de apresentação dos elementos;
- ✓ Tabindex;
- ✓ Tipos de mídia.

9.1.Introdução

A transformação de um layout em formato de imagem criado por um designer para um formato HTML que servirá um site ou Web App é um processo que requer alguns passos. Nesta leitura, abordaremos como criar um layout e posicioná-lo corretamente em um navegador, de maneira que envolva as folhas de estilo em cascata (CSS) e os elementos que aprendemos até aqui.

Um dos métodos utilizados para a criação de layout em HTML é o chamado **Tableless**, que é o nome atribuído à metodologia de construção de sites sem o uso de **tables** (tabelas). Para realizar essa construção, utiliza-se o HTML para determinar a estrutura dos dados e as folhas de estilo CSS para formatar sua exibição.

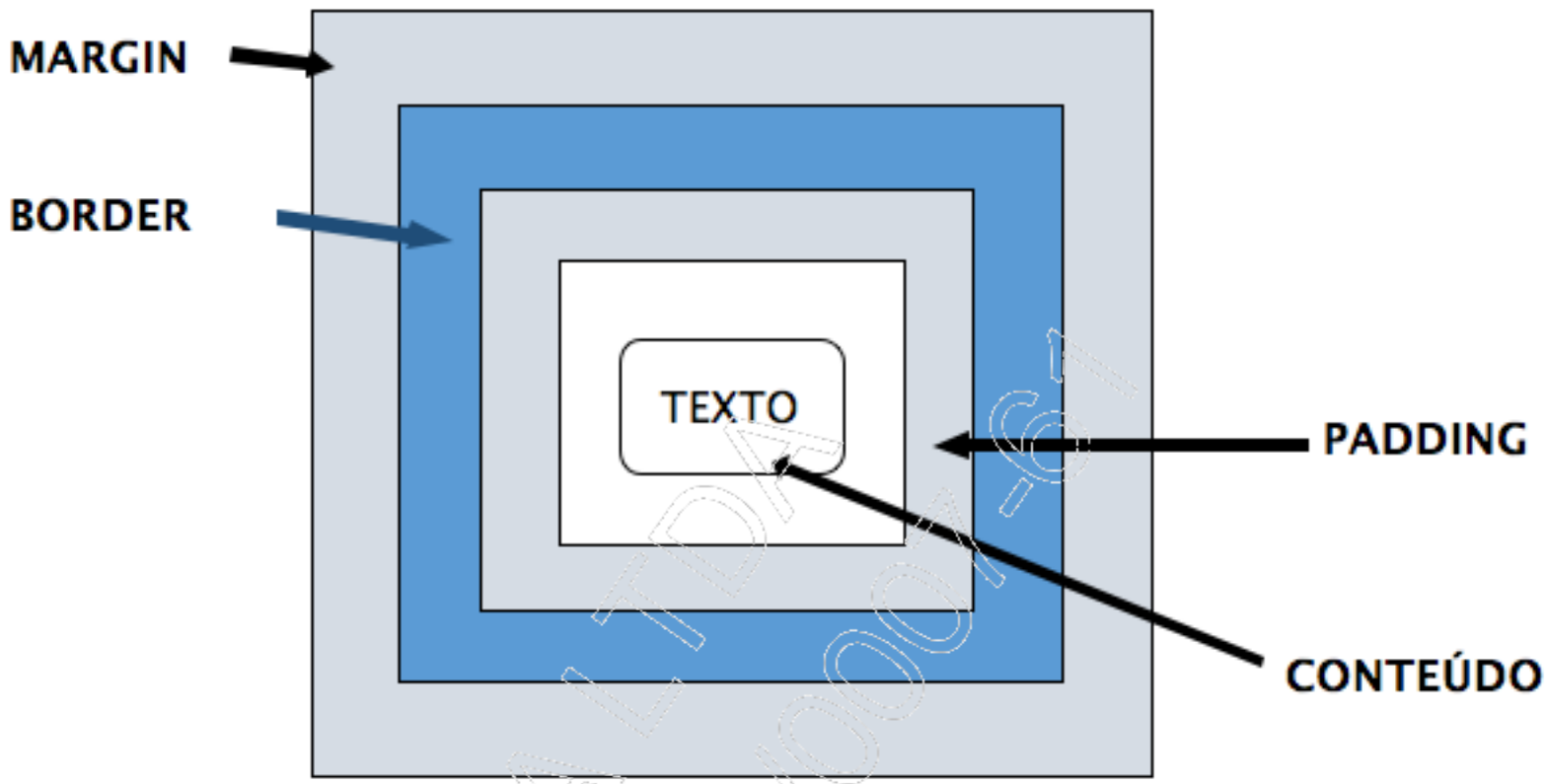
Antigamente, era comum o uso de tabelas para a estruturação e criação do layout da página, porém, desde o XHTML não há necessidade de utilizá-las.

Devemos ter em mente que as tags, inclusive a **<table>**, continuam a ser utilizadas de acordo com suas funções. A construção de sites por meio da metodologia **Tableless** requer a formatação de elementos por meio de arquivos CSS, que representam as folhas de estilo.

9.2.Dimensão real dos elementos

A dimensão real ocupada por um elemento dentro de uma página deve ser conhecida para que se possa compreender os conceitos envolvidos no processo de construção de um layout. A área ocupada por cada um dos elementos de uma página é denominada contêiner e tem a forma de um retângulo.

O contêiner de um elemento possui todos os itens que dizem respeito a ele, como seu conteúdo, seus espaços em branco, suas margens e suas linhas de contorno. O desenho a seguir demonstra quais são as áreas de um contêiner de um elemento. Observe:



Em que:

- **MARGIN**: Espaço que separa o contêiner de outros elementos que compõem a página. Vale destacar que a margem (margin) não está dentro dos limites de um elemento. As margens são utilizadas com a finalidade de mover a caixa do elemento;

- **CONTEÚDO**: Todos os itens contidos em um elemento, os quais permanecem dentro do contêiner;

- **BORDER**: Limites de um elemento, isto é, as linhas de contorno do contêiner;

- **PADDING**: Espaços em branco existentes entre o conteúdo e os limites de um elemento.

A largura do contêiner, no qual está contido o elemento, é determinada pela largura do conteúdo somada às larguras referentes ao **border** e ao **padding**. Já a largura do elemento em si é determinada apenas pela largura ocupada por seu conteúdo, assim como sua altura.

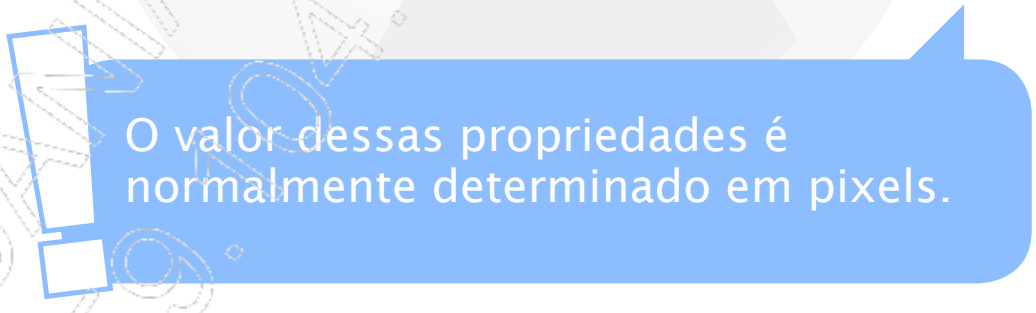
Em uma página, todos os elementos visíveis são de grupo ou **inline**. Os elementos de bloco têm sua largura determinada pela largura do bloco em que estão. Eles iniciam-se sempre em uma linha nova e, depois de finalizados, há uma nova mudança de linha. Já os elementos **inline** têm sua largura determinada apenas por seu conteúdo e, ao contrário dos elementos de bloco, não se iniciam sempre em uma nova linha. Dessa forma, concluímos que os elementos **inline** comportam-se da mesma maneira que um texto simples.

Como exemplo de elemento de bloco, podemos mencionar o **<table>**. Já como exemplo de um elemento **inline**, podemos mencionar o ****.

9.3.Configurando as margens dos elementos

Em CSS, a espessura das margens dos elementos é definida pelas propriedades dessas margens, as quais podem ser:

- **margin-bottom**: Permite determinar a espessura referente à margem inferior do elemento;
- **margin-top**: Permite determinar a espessura referente à margem superior do elemento;
- **margin-right**: Permite determinar a espessura referente à margem direita do elemento;
- **margin-left**: Permite determinar a espessura referente à margem esquerda do elemento.



Observe o exemplo a seguir, em que temos a definição das margens de um elemento:

- **style.css**

```
p {
    margin-top: 70px;
    margin-left: 50px;
}
```

- **aula9_1.html**

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <p>
                Testando o atributo
                margin-top.
            </p>
        </main>
    </body>
</html>
```

Ao executarmos esse código, o resultado obtido é o seguinte:

9.4.Posicionando os elementos

Quando trabalhamos com arquivos CSS, podemos posicionar elementos e construir o layout com algumas propriedades. Duas propriedades do CSS3 que serão utilizadas com o objetivo de criar e posicionar um layout são **Flexible Box Layout**, ou simplesmente **Flexbox**, que é uma forma simples e sem cálculos para posicionar um layout, e o **Módulo Grid Template Layout**, ainda sendo padronizado em 2013.

Além disso, temos outras duas propriedades que já eram utilizadas no XHTML: **float** e **position**. Neste tópico, veremos como trabalhar com elas.

9.4.1.Position

A propriedade **position** permite determinar o posicionamento de um conteúdo na tela do usuário. Esse posicionamento pode ser: **estático**, **fixo**, **absoluto** e **relativo**.

9.4.1.1. Posicionamento estático

Este tipo de posicionamento é determinado por meio do valor **static**, o qual é desnecessário, pois o posicionamento estático é padrão para os elementos. Portanto, a propriedade **position**: **static** não precisa ser declarada de forma explícita na folha de estilo.

Um elemento cuja posição é estática coloca-se imediatamente abaixo de seu elemento anterior e acima de seu posterior. No entanto, vale destacar que isso apenas ocorre caso os elementos anterior e posterior não tenham seu posicionamento determinado de forma diferente da estática.

9.4.1.2. Posicionamento fixo

Este tipo de posicionamento determina que o elemento seja posicionado de acordo com a parte que pode ser visualizada do **User Agent**, no qual é exibida a página. Para determinar que um elemento terá seu posicionamento fixo, é preciso utilizar o valor **fixed** juntamente à propriedade **position**.

9.4.1.3. Posicionamento absoluto

Este tipo de posicionamento determina que o elemento seja posicionado levando-se em consideração o local em que está o elemento mais próximo, cuja posição pode ser definida como **fixa**, **absoluta** ou **relativa**.

Nas situações em que não há um elemento mais próximo, considera-se o posicionamento do elemento **<body>.** Para determinar o posicionamento de um elemento como sendo **absoluto**, basta utilizar o valor **absolute** em conjunto à propriedade **position**.

Vale destacar que há quatro propriedades que se aplicam aos elementos cujo posicionamento é determinado como **absoluto**. São elas: **bottom**, **top**, **left** e **right**.

Observe o exemplo a seguir:

- **style.css**

```css
#imagem {
    position: absolute;
    left: 30px;
    top: 60px
}
```

- **aula9_2.html**

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <img src="images/logotipo.png" alt="Logo HTML5" id="imagem">
        </main>
    </body>
</html>
```

Ao executarmos o código anterior, o resultado obtido é o seguinte:

Podemos observar, no exemplo apresentado, a presença da propriedade **position**, a qual recebe o valor **absolute** e é complementada pelas propriedades **left** e **top**. Com isso, o posicionamento do elemento é determinado tomando-se como base o canto superior esquerdo da viewport. O elemento é representado neste exemplo por **img**, que possui a **id=”imagem”**.

Podemos observar, ainda, que a propriedade **left** determina o distanciamento da imagem a partir da margem esquerda e a propriedade **top** determina o distanciamento a partir da margem superior.

9.4.1.4. Posicionamento relativo

Este tipo de posicionamento determina a posição do elemento de acordo com sua própria posição. Para determinar o posicionamento de um elemento como **relativo**, basta utilizar o valor **relative** juntamente à propriedade **position**.

Quando trabalhamos com o posicionamento relativo de elementos, podemos utilizar as propriedades **bottom**, **top**, **right** e **left**.

Observe o exemplo a seguir:

- **style.css**

```css
.imagem {
    position: relative;
    left: 30px;
    top: 60px
}
```

- **aula9_3.html**

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <img src="images/logotipo.png" alt="Logo HTML5" class="imagem">
            <img src="images/css3.png" alt="Logo css3" class="imagem">
        </main>
    </body>
</html>
```

Veja o resultado:

9.4.2. Float

Os elementos que são declarados como **float** são convertidos para elementos de bloco de forma automática, podendo ter sua altura e largura declaradas. Mais especificamente, a largura dos elementos **float** deve ser declarada, pois os browsers consideram a não declaração dessa largura um erro.

Quanto ao posicionamento de um elemento **float**, este ocorre imediatamente após o elemento em bloco anterior a ele, embora tais elementos fiquem fora do fluxo. Os valores que podem ser utilizados juntamente ao atributo **float** são: **none**, **left**, **right** e **inherit**, cujo valor é igual ao valor da propriedade referente ao elemento pai.

Veja o exemplo a seguir:

- **style.css**

```css
img {
    float: left;
    width: 15%;
}
p {
    font-size: 20px;
    margin-top: 0px
}
```

- **aula9_4.html**

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <img src="images/logotipo.png" alt="Logo HTML5" class="imagem">
            <p>Texto principal</p>
        </main>
    </body>
</html>
```

Veja o resultado obtido:

Se não tivéssemos utilizado o atributo **float** no exemplo anterior, o resultado obtido seria o seguinte:

9.5.Definindo as camadas

Alguns elementos de bloco, como **section**, **div**, **article** e outros mencionados anteriormente, oferecem atributos que permitem definir as camadas que serão utilizadas na construção do layout da página.

Veja, a seguir, quais são esses atributos:

- **id**: Permite determinar como a camada será identificada;
- **class**: Permite determinar qual classe já declarada no arquivo CSS deve ser aplicada;
- **z-index**: Permite determinar o nível de empilhamento;
- **visibility**: Permite determinar a visibilidade da camada. Os valores que podem ser utilizados por este atributo são os seguintes: **visible**, **inherit** e **hidden**.

O atributo **z-index**, que tem a finalidade de ordenar a prioridade de visualização dos elementos, funciona apenas com os elementos que foram configurados com o posicionamento **absoluto**, o que significa que são os elementos cuja propriedade **position** é configurada como **absolute**. Vale destacar que quanto maior o valor do atributo **z-index**, maior a prioridade de visualização do elemento. Por exemplo, um elemento que possui **z-index**: 5 tem prioridade sobre um elemento que possui **z-index**: 3.

Observe o exemplo a seguir:

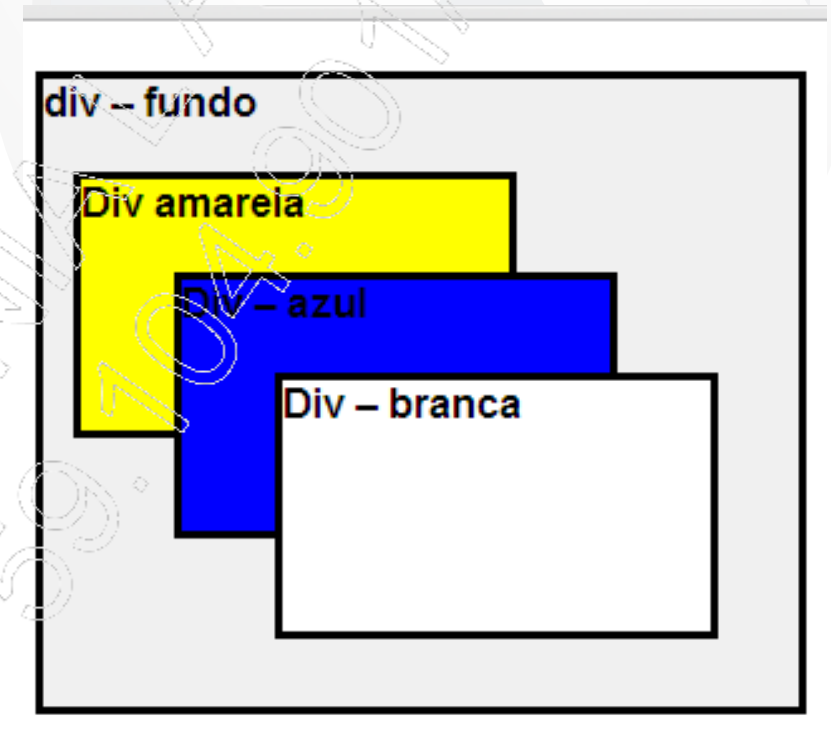
- **style.css**

```
.posicao{
    position: absolute;
    width:170px; height: 100px;
    font-family: arial;
    font-size: 16px;
    font-weight: bold;
}
#branca{
    top:140px;left:100px;
    background-color: #fff;
    border: solid 3px #000;
    z-index: 4;
}
#azul{
    top:100px;left:60px;
    background-color: #00f;
    border: solid 3px #000;
    z-index: 3;
}
#amarela{
    top:60px;left:20px;
    background-color: #ff0;
    border: solid 3px #000;
    z-index: 2;
}
#fundo{
    top:20px;left:5px;
    width:300px;height: 250px;
    background-color: #eee;
    border: solid 3px #000;
    z-index: 1;
}
```

- **aula9_5.html**

```
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <div id="amarela">Div amarela</div>
            <div id="azul">Div - azul </div>
            <div id="branca">Div - branca</div>
            <div id="fundo">div - fundo</div>
        </main>
    </body>
</html>
```

O resultado produzido será o seguinte:



Neste exemplo, temos o posicionamento de várias camadas. Com isso, algumas sobreposições foram criadas.

9.6.Definindo o modo de apresentação dos elementos

O atributo **display** permite determinar se um elemento deve ser apresentado e, caso deva, permite determinar seu modo de apresentação. Os valores que podem ser utilizados com o atributo **display** são os seguintes:

Valores	Descrição
table	Determina que o elemento seja apresentado como uma tabela, sendo que há mudança de linha tanto antes do elemento quanto depois dele.
list-item	Determina que o elemento seja apresentado com sendo item de uma lista.
inline	Determina que o elemento seja apresentado sem que haja mudança de linha.
block	Determina que o elemento seja apresentado como um elemento de bloco, e, diferentemente do item anterior, existe a quebra de linha antes e depois do item.
none	Determina que o elemento não seja apresentado.

O exemplo descrito a seguir permite compreender de forma adequada a utilização dos valores que acabam de ser definidos:

- **style.css**

```css
.linha{display: inline;}
```

- **aula9_6.html**

```html
<!doctype HTML>
<html>
    <head>
        <meta charset="utf-8" />
        <title>HTML5 Fundamentos - CSS</title>
        <link type="text/css" href="css/style.css"rel="StyleSheet" />
    </head>
    <body>
        <main role="main">
            <p class="linha"> PARAGRAFO 1 * PARA TESTE DO ATRIBUTO DISPLAY </p>
            <p class="linha"> PARAGRAFO 2* PARA TESTE DO ATRIBUTO DISPLAY </p>
            <p class="linha"> PARAGRAFO 3* PARA TESTE DO ATRIBUTO DISPLAY </p>
        </main>
    </body>
</html>
```

O resultado produzido será o seguinte:

```
PARAGRAFO 1 * PARA TESTE DO ATRIBUTO DISPLAY PARAGRAFO 2* PARA TESTE DO ATRIBUTO DISPLAY
PARAGRAFO 3* PARA TESTE DO ATRIBUTO DISPLAY
```

9.7.Tabindex

Por meio desse atributo, podemos atribuir uma ordem de navegação ao longo de todos os elementos existentes em um documento HTML5, como links e elementos de formulário. Sem o uso de **tabindex**, o usuário seria obrigado a percorrer todos esses elementos de acordo com a ordem em que eles aparecem no código.

É possível determinar o índice de tabulação em uma barra de navegação, em elementos pertencentes a um formulário ou outros itens que fazem parte do código XHTML. Para que o índice de tabulação seja definido em um campo de formulário e um link, por exemplo, devemos utilizar a sintaxe a seguir:

```
<input type="email" id="login" tabindex="1" />
```

9.8.Tipos de mídia

Alguns estilos podem ser aplicados somente em algumas situações especiais, por exemplo, a utilização de um determinado tipo de fonte assim que algum usuário requisitar a impressão da página. Por meio dos tipos de mídia (também chamados de media types), um arquivo CSS poderá ser criado apenas para um evento em particular.

Veja quais são os principais tipos de mídia para aplicação em uma folha de estilo CSS:

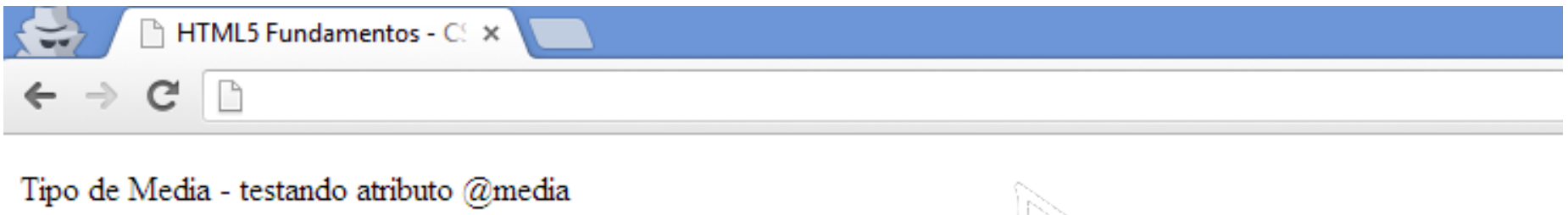
- **all**: Permite a definição de estilos para todos os tipos de mídia;
- **aural**: Os estilos são definidos para sintetizadores de voz;
- **braille**: Possibilita a aplicação de definições para textos em braile;
- **embossed**: Permite a impressão em impressoras próprias para a leitura braile;
- **handheld**: Os estilos são definidos para equipamentos móveis de mão, como celulares;
- **print**: Possibilita a aplicação de estilos para a impressão da página em papel;
- **projection**: Os estilos são definidos para a apresentação da página em um projetor;
- **screen**: Permite que os estilos sejam utilizados para que a página seja exibida na tela do computador;
- **tty**: Os estilos são aplicados para a apresentação da página em terminais com poucos recursos;
- **tv**: Permite a exibição da página em uma tela de TV.

Por meio da regra **@media**, é possível aplicar diversas propriedades relacionadas a diferentes tipos de mídia em uma só folha de estilo. No exemplo a seguir, temos uma demonstração dessa regra: a página será formatada para impressão em papel e exibição na tela. Para começar, será criado o arquivo **.css**:

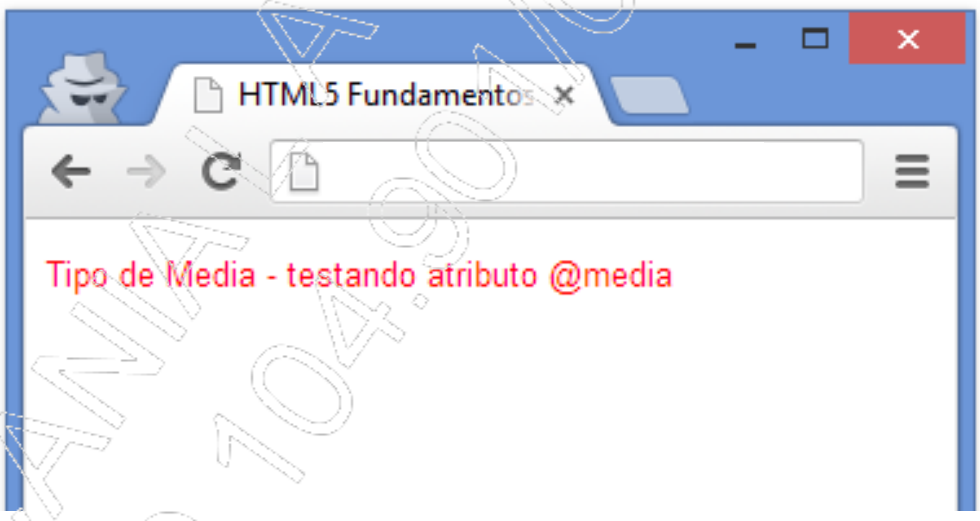
```
@media print {
 .midia {
 font-family: arial;
 font-size: 11px;
 color: black;
 text-align: left
 }
}
@media only screen and (max-width: 480px){
 .midia {
 font-family: arial;
 font-size: 13px;
 color: red;
 text-align: left
 }
}
```

No exemplo anterior, temos um estilo com cor preta para impressora e outro estilo para dispositivos que possuem no máximo 480px de largura.

Para realizar este teste podemos utilizar o redimensionar do navegador. Observe que com a janela completamente dimensionada, ultrapassamos o valor de 480px de largura, logo, o estilo não será aplicado:

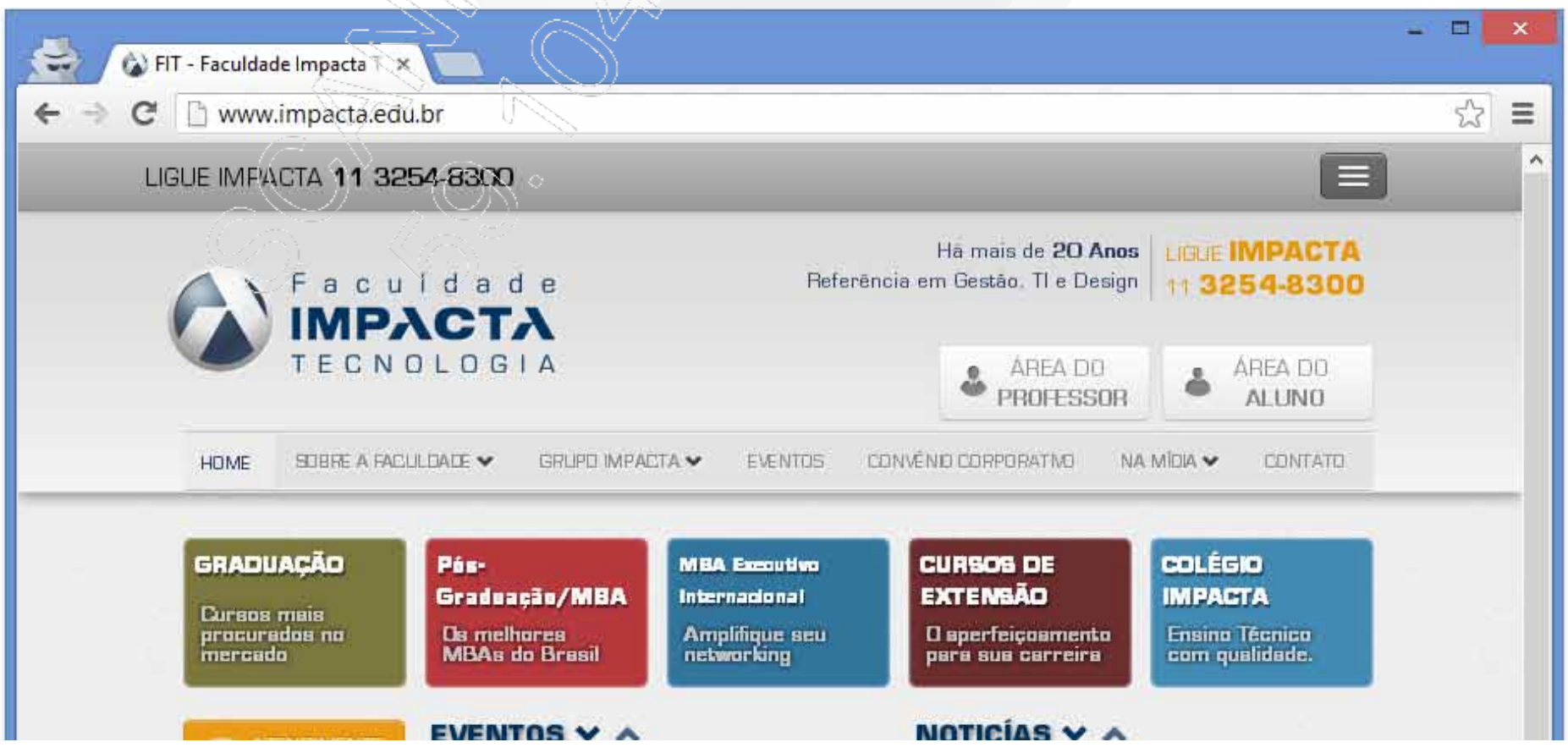


Agora, quando redimencionamos o navegador para uma resolução inferior a 480px o estilo é aplicado:



O layout do site da Faculdade Impacta Tecnologia utiliza esse recurso, que é denominado Responsive Web Design ou apenas Layout Responsivo:

Reduzindo a dimensão da janela, por exemplo, no uso em tablets, percebemos que o banner principal sumirá:



E com a tela utilizada para SmartPhone:

Teste seus conhecimentos

9 Criando e posicionando um layout

1. Qual propriedade devemos utilizar para determinar a margem interna de um elemento?

- ☐ a) internal-margin
- ☐ b) padding
- ☐ c) border
- ☐ d) margin-internal
- ☐ e) padding-border

2. Qual propriedade e valor devemos utilizar para determinar um posicionamento de um elemento de forma absoluta?

- ☐ a) float:none
- ☐ b) float:left
- ☐ c) position:relative
- ☐ d) position:absolute
- ☐ e) position:absolute z-index 3

3. Qual propriedade da CSS utilizamos para determinar a prioridade de exibição de um elemento em camada?

- ☐ a) id
- ☐ b) class
- ☐ c) z-index
- ☐ d) visibility
- ☐ e) inline

4. Qual propriedade e valor devemos utilizar para ocultar um elemento?

- ☐ a) border:none
- ☐ b) top:none
- ☐ c) display:block
- ☐ d) display:inline
- ☐ e) display:none

Mãos à obra!

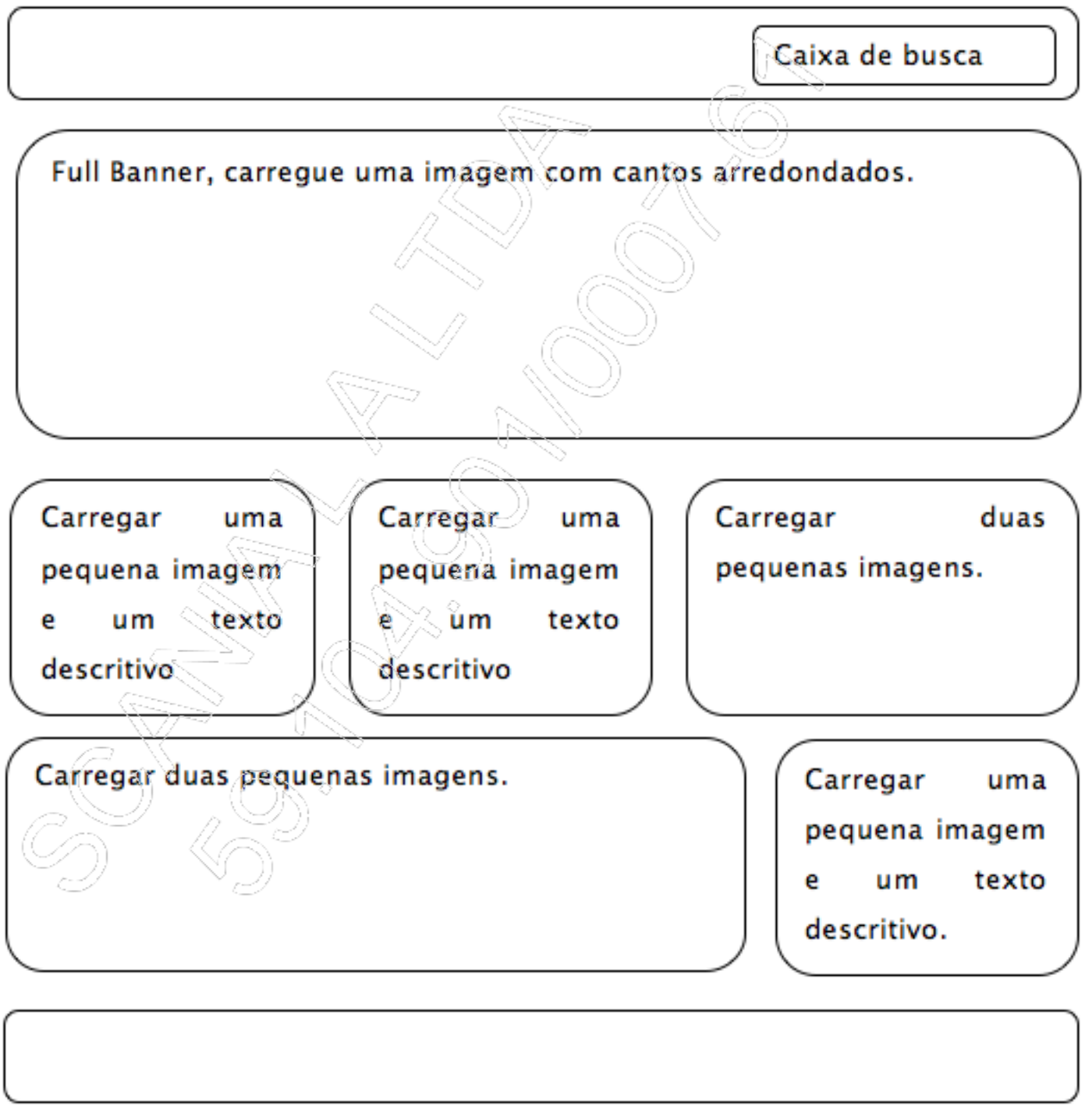
Criando e posicionando um layout

9

Laboratório 1

A – Criando um layout

1. Utilizando o wireframe a seguir, crie um arquivo chamado **home.html** e um arquivo para folha de estilos denominado **estilo.css** de forma que produza o seguinte resultado:



Formulários

10

- ✓ Atributos para controle de formulários;
- ✓ Atributos para validar formulários;
- ✓ Atributo type (elemento input).

10.1.Introdução

Em HTML, um formulário é uma seção de documento que abrange texto, marcação, controles e rótulos. A função de um formulário é recolher dados recolhidos pelo usuário e enviá-los para serem processados no servidor. Os dados são inseridos pelo usuário nos formulários por meio de modificações nos controles (digitando um texto, selecionando um item em um menu, etc.) e depois processados por scripts no servidor.

Como, ao preencher um formulário, um usuário pode inserir não só textos simples, mas também scripts inteiros, isso pode gerar um problema de segurança, com a entrada de scripts maliciosos que prejudicam o site e seus usuários. Por isso, ao desenvolver um formulário e o script para o seu processamento, o mecanismo de validação é importante. A validação é o que permite verificar os dados digitados por um usuário nos controles de um formulário.

Até a HTML5, a validação era feita com JavaScript, mas podia ser facilmente anulada. Na HTML5, a validação é simples e eficiente, reduzindo a necessidade de desenvolvimento de scripts. Há novos atributos nos controles que permitem definir o tipo de dado a ser entrado e disparam mecanismos nativos de validação e mensagens de erro.

Nesta leitura, você verá as funcionalidades para formulários na HTML5, implementadas por meio de atributos usados nos controles dos formulários.

10.2.Atributos para controle de formulários

As funcionalidades para formulários na HTML5 estão relacionadas a novos atributos usados nos controles. Como veremos, eles servem para validação e outras funcionalidades que antes eram implementadas com o uso de scripts.

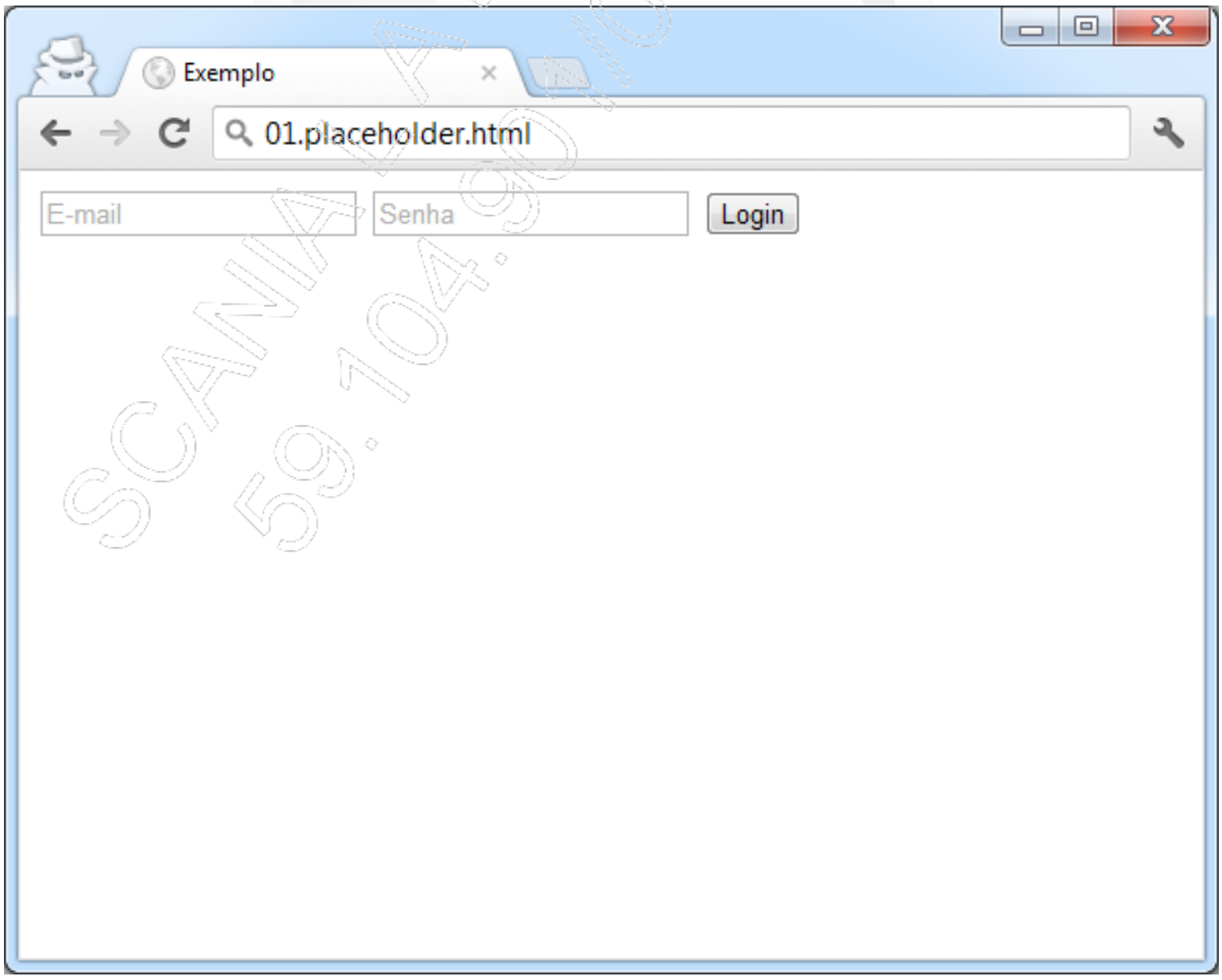
10.2.1.placeholder

Sua função é definir uma dica de preenchimento de um campo, composta por uma palavra ou uma frase curta. Deve ser usado exclusivamente com os elementos **input** e **textarea**.

A seguir, um exemplo ilustrando o uso de **placeholder**:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form>
10         <input type="text" placeholder="E-mail">
11         <input type="password" placeholder="Senha">
12         <button type="submit" class="btn">Login</button>
13     </form>
14
15 </body>
16 </html>
```

Esse código produzirá o seguinte resultado:



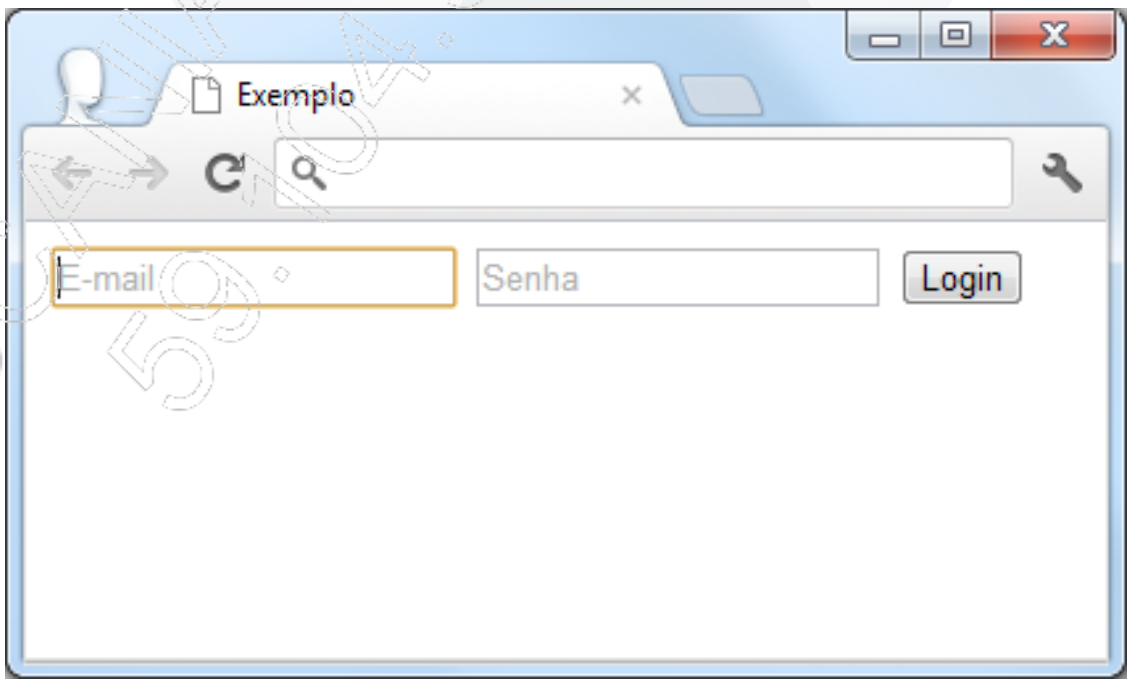
10.2.2.autofocus

A função deste atributo é definir qual o elemento deve ter o foco quando a página for carregada. Pode ser usado com os elementos **button**, **input**, **keygen**, **select** e **textarea**.

Veja o uso de **autofocus** no seguinte exemplo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form>
        <input autofocus type="text" placeholder="E-mail">
        <input type="password" placeholder="Senha">
        <button type="submit" class="btn">Login</button>
    </form>

</body>
</html>
```

O resultado será o seguinte:



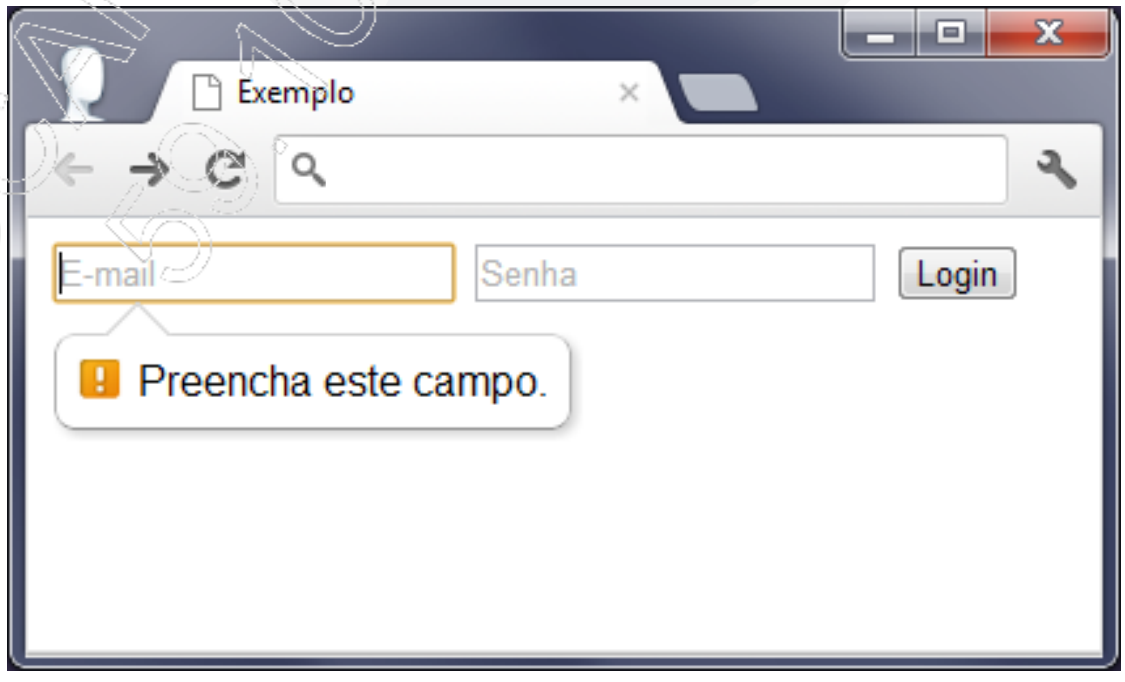
10.2.3.required

Sua função é marcar um controle de preenchimento obrigatório. Pode ser usado com **input**, **select** e **textarea** e é um atributo booleano, ou seja, basta sua presença na sintaxe, sem declaração de valor.

Veja, a seguir, um exemplo de **required**:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form>
        <input required type="text" placeholder="E-mail">
        <input required type="password" placeholder="Senha">
        <button type="submit" class="btn">Login</button>
    </form>

</body>
</html>
```

Dê uma olhada no resultado:



10.2.4.autocomplete

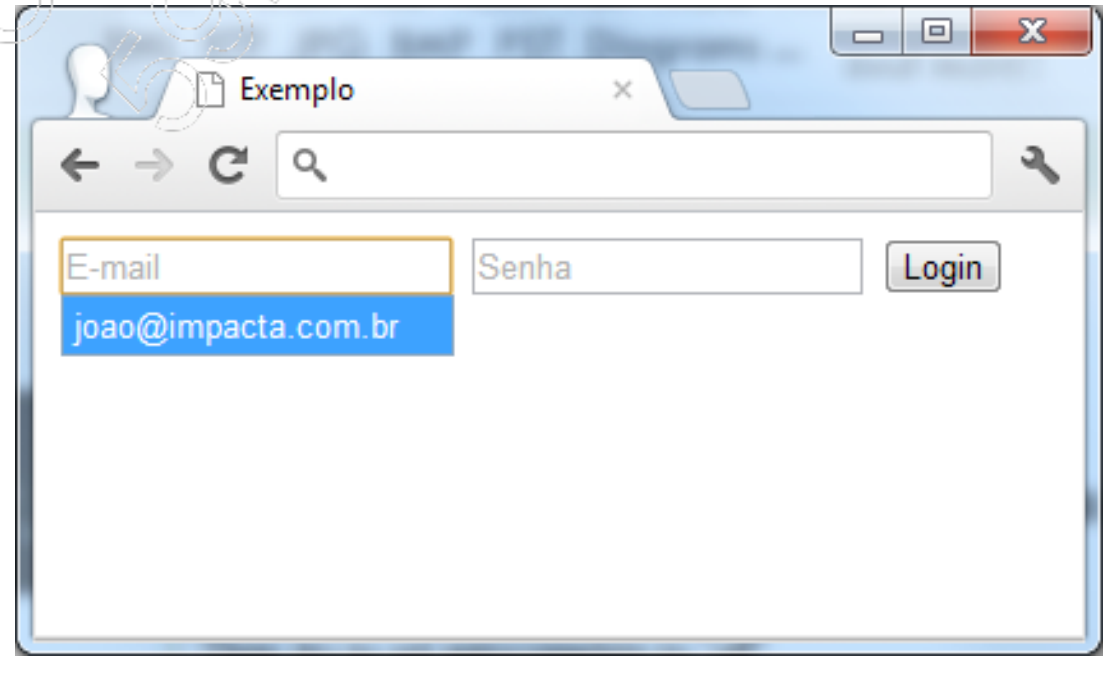
Este atributo oferece uma lista de opções para o preenchimento de um controle, de acordo com o que o usuário digitou nele anteriormente. É implementado de diferentes maneiras nos navegadores. Pode ser usado com os elementos **form** e **input** e admite os seguintes valores: **on** e **off**. Eles definem, respectivamente, se o navegador fará ou não sugestões de autopreenchimento.

Quando não é especificado para um controle específico, ele assume o comportamento de autopreenchimento do elemento **form** ao qual o controle pertence. Quando o **autocomplete** não é especificado para o **form**, o padrão de autopreenchimento é **on**.

Veja um exemplo de **autocomplete**:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form autocomplete="on">
        <input type="text" name="email" placeholder="E-mail">
        <input type="password" placeholder="Senha">
        <button type="submit" class="btn">Login</button>
    </form>

</body>
</html>
```

O código produzirá o seguinte resultado:

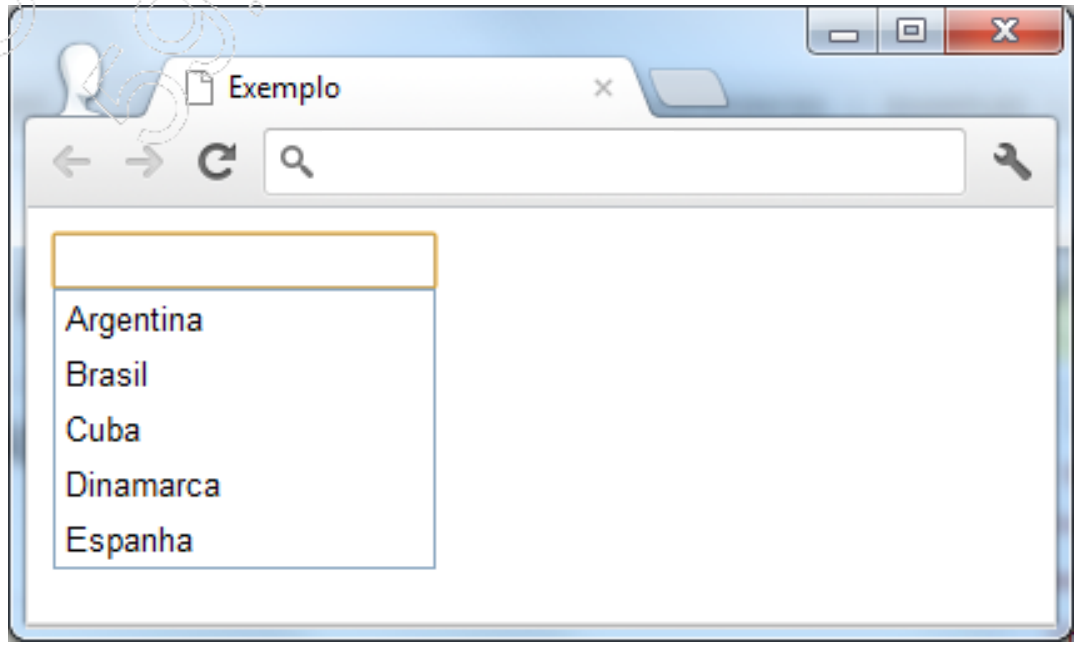


10.2.5.list

Define uma lista com opções de preenchimento para um campo de texto. Deve ter o mesmo valor que o id de um elemento **datalist** e só pode ser usado com o elemento **input**. Veja um exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form>

        <input list="paises" />

        <datalist id="paises">
          <option value="Argentina">
          <option value="Brasil">
          <option value="Cuba">
          <option value="Dinamarca">
          <option value="Espanha">
        </datalist>

    </form>

</body>
</html>
```

O resultado será este:



10.2.6. max

Este atributo tem como função definir um valor máximo para os diferentes elementos com os quais pode ser usado:

- Para o elemento **meter**, define o valor máximo da escala da medida marcada. Caso seja omitido, seu valor é 1;

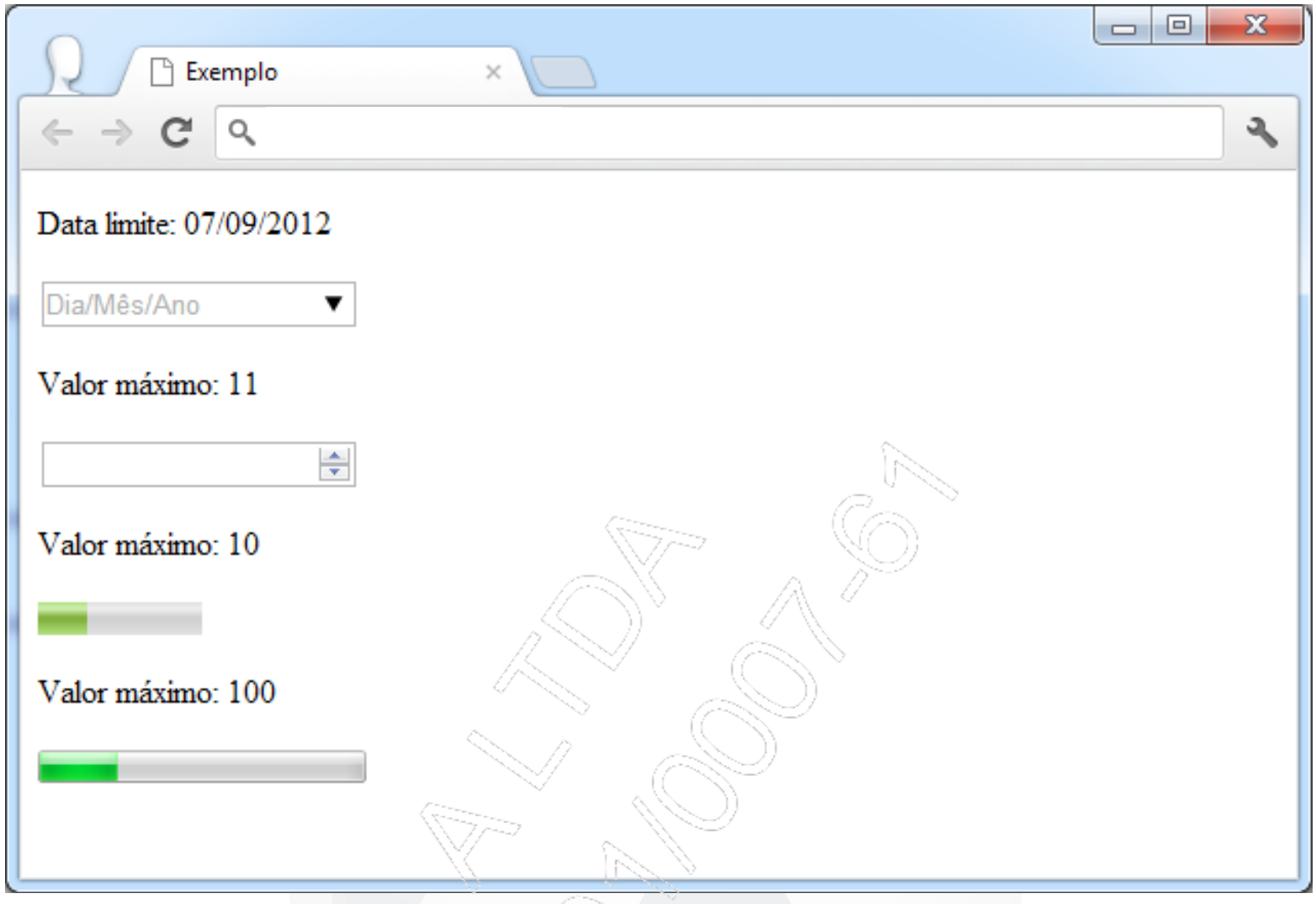
- Para o elemento **progress**, define o valor máximo de uma barra de progresso, que mostra a progressão de uma tarefa;

- Para o elemento **input**, define o valor máximo que será permitido como entrada em um campo. Se for usado junto com o atributo **min**, definirá uma faixa de entrada de dados no campo.

Veja, a seguir, um exemplo de como utilizar o atributo **max**:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form>

        <p>Data limite: 07/09/2012</p>
        <input type="date" max="2012-09-07"><br/>

        <p>Valor máximo: 11</p>
        <input type="number" max="11" /><br/>

        <p>Valor máximo: 10</p>
        <meter value="3" max="10">3 de 10</meter><br/>

        <p>Valor máximo: 100</p>
        <progress value="25" max="100"></progress>

    </form>

</body>
</html>
```

E agora observe o resultado:



10.2.7.min

Ao contrário de **max**, serve para definir um valor mínimo. Além disso, só pode ser usado com **meter** e **input**:

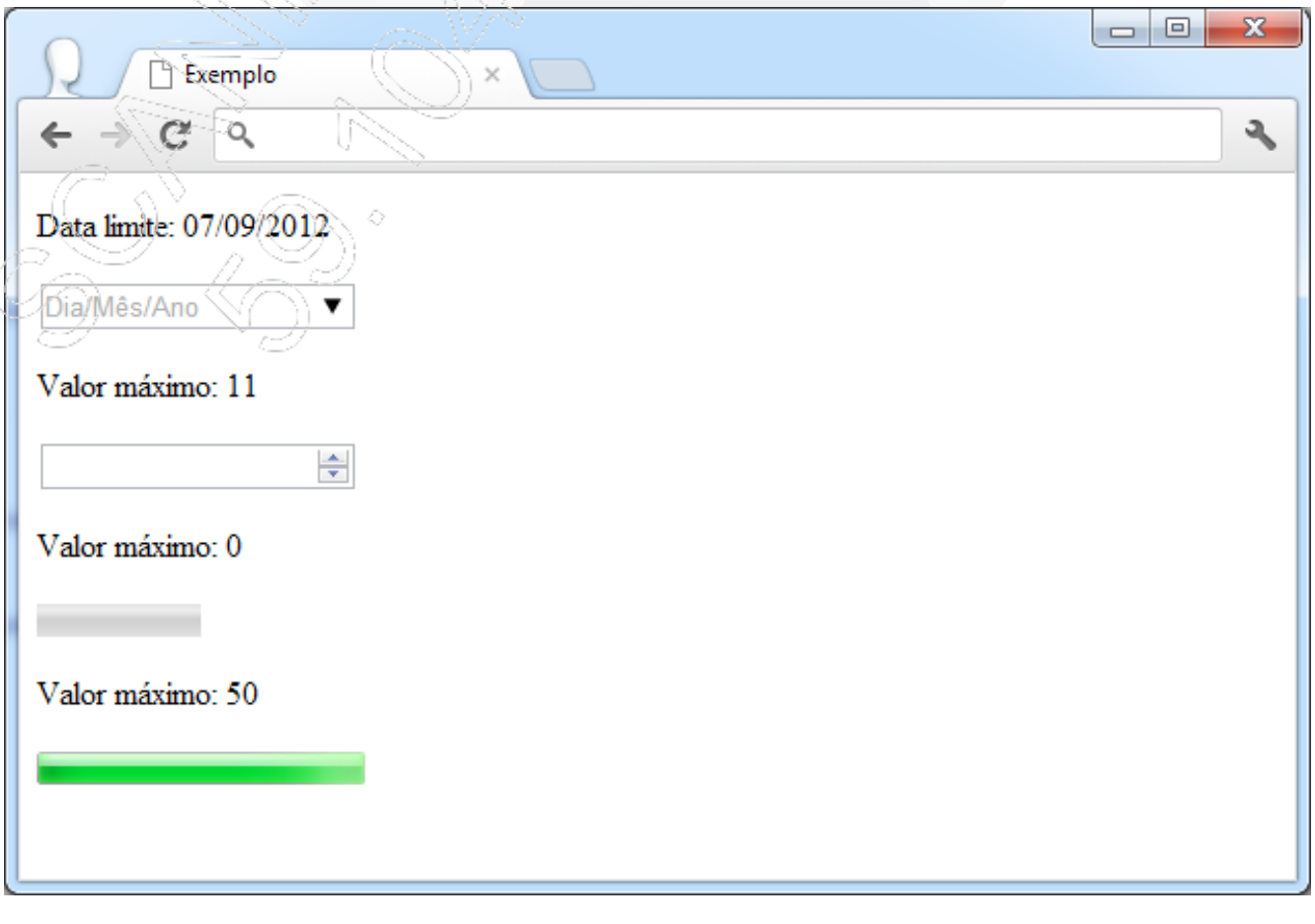
- Para **meter**, define o valor mínimo da escala da medida marcada. Caso seja omitido, seu valor é 0;

- Para **input**, define o valor mínimo que será permitido como entrada em um campo.

Veja, no exemplo, como utilizar **min**:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form>

        <p>Data limite: 07/09/2012</p>
        <input type="date" min="2012-09-07"><br/>

        <p>Valor minimo: 11</p>
        <input type="number" min="11" /><br/>

        <p>Valor minimo: 0</p>
        <meter value="1" min="0">3 de 10</meter><br/>

        <p>Valor minimo: 50</p>
        <progress value="50" min="50"></progress>

    </form>

</body>
</html>
```

O resultado você vê a seguir:



10.2.8. form

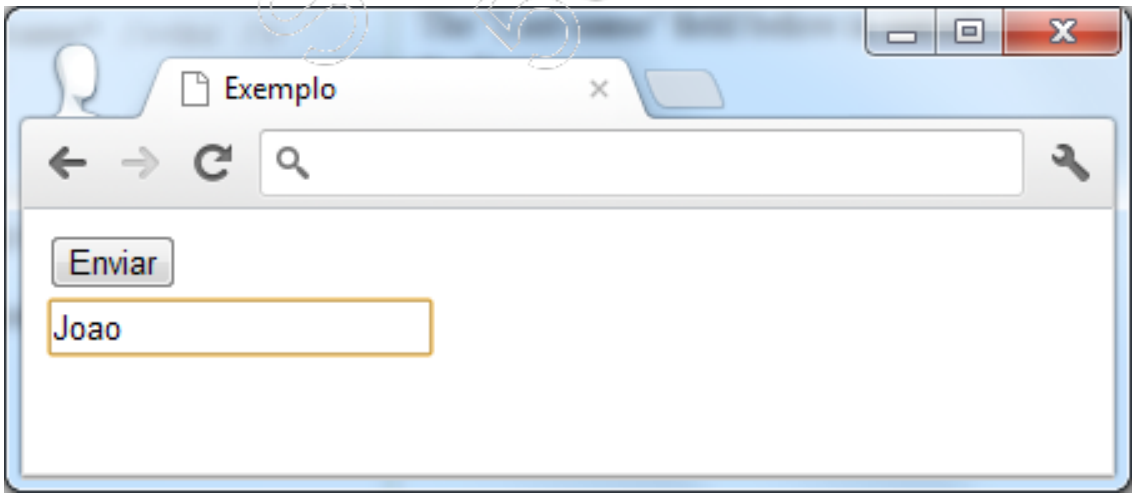
Associa um controle de formulário a um elemento formulário. Permite, inclusive, associar a um formulário controles que estão fora dele ou dentro de outros formulários, o que não era possível nas versões anteriores da HTML. O atributo **form** tem o mesmo valor do atributo **id** do formulário ao qual estiver associado.

Podemos usar **form** com os seguintes elementos: **input**, **output**, **select**, **object**, **button**, **textarea**, **meter**, **label**, **progress**, **keygen** e **fieldset**.

Veja como usar o atributo no exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form id="formulario" action="#">
        <input type="submit" value="Enviar">
    </form>

    <input type="text" name="nome" placeholder="Nome" form="formulario">

</body>
</html>
```

E este será o resultado:



Observe, na imagem a seguir, que, mesmo o elemento estando fora da tag do formulário, o seu valor é enviado:

10.2.9. formaction

Este atributo define uma URL que será o endereço para o qual o formulário será enviado e onde será processado. Pode ser usado com os elementos **input** e **button**. Veja como usá-lo no exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="login_aluno.html">
        <input type="text" placeholder="E-mail">
        <input type="password" placeholder="Senha">
        <input type="submit" value="Login Aluno">
        <input type="submit" value="Login Instrutor" formaction="login_instrutor.html">
    </form>

</body>
</html>
```

10.2.10.formenctype

Este atributo define qual é o tipo de conteúdo enviado pelo formulário e pode ser usado com **input** e **button**. Veja como usá-lo no exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#" method="post">
        <input type="text" placeholder="Nome">
        <input type="submit" value="Envio Padrão">
        <input type="submit" value="Envio Texto Plano" formenctype="text/plain">
    </form>

</body>
</html>
```

10.2.11.formmethod

A função deste atributo é definir qual método será usado para enviar os dados do formulário. Pode ser usado com **input** e **button**. Veja como no seguinte exemplo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#" method="post">
        <input type="text" name="nome" placeholder="Nome">
        <input type="submit" value="POST">
        <input type="submit" value="GET" formmethod="get">
    </form>

</body>
</html>
```

10.2.12.formtarget

Permite determinar qual será o destino do documento aberto após o formulário ser enviado. Pode ser usado com **input** e **button** e admite os seguintes valores: **_blank**, **_top**, **_self**, **_parent** ou outro nome na janela atual.

O exemplo a seguir mostra como usar este atributo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <input type="text" name="nome" placeholder="Nome">
        <input type="submit" value="Enviar Padrão">
        <input type="submit" value="Enviar Para Nova Janela" formtarget="_blank">
    </form>

</body>
</html>
```

10.3.Atributos para validar formulários

Como dissemos, uma característica importante da HTML5 é introduzir atributos que facilitam a validação de formulários, automatizando o processo todo ou pelo menos parte dele. Vejamos os atributos que são utilizados nessa tarefa.

10.3.1.formnovalidate

Com este atributo, você desabilita a validação de dados do formulário. Pode ser usado com os elementos **button** e **input** e é um atributo booleano. Assim, basta inserir o atributo em um elemento para que o formulário não tenha seus dados validados. Veja no exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <input required type="text" name="nome" placeholder="Nome">
        <input type="submit" value="Enviar com Validação">
        <input type="submit" value="Enviar sem Validação" formnovalidate="formnovalidate">
    </form>

</body>
</html>
```

10.3.2.novalidate

Possui o mesmo efeito que o **formnovalidate**. A diferença é que deve ser usado exclusivamente com um elemento **form**. Veja um exemplo com esse atributo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#" novalidate="novalidate">
        <input required type="email" name="nome" placeholder="Nome">
        <input type="submit" value="Enviar">
    </form>

</body>
</html>
```

10.3.3. pattern

Com **pattern**, você pode definir expressões regulares para validação de formulários, sem precisar de JavaScript.

Veja como utilizá-lo no exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <label for="CPF_1">Digite o CPF com máscara:</label>
        <input
            id="CPF_1"
            type="text"
            pattern="\d{3}\.\d{3}\.\d{3}-\d{2}"
            placeholder="___.___.___-__">
        <br/>
        <label for="CPF_2">Digite o CPF sem máscara:</label>
        <input
            id="CPF_2"
            type="text"
            pattern="[0-9]{11}"
            placeholder="Apenas números">
        <br/>
        <input type="submit" value="Enviar">
    </form>

</body>
</html>
```

10.4.Atributo type (elemento input)

Na HTML5, o atributo **type** do elemento **input** possui diversos novos valores que definem o tipo de dado esperado pelo controle. Na HTML4, o **type** tinha dez valores diferentes (**text**, **password**, **checkbox**, **radio**, **submit**, **reset**, **file**, **hidden**, **image** e **button**). Com a HTML5, esses valores foram mantidos e foram acrescentados outros, que você verá adiante.

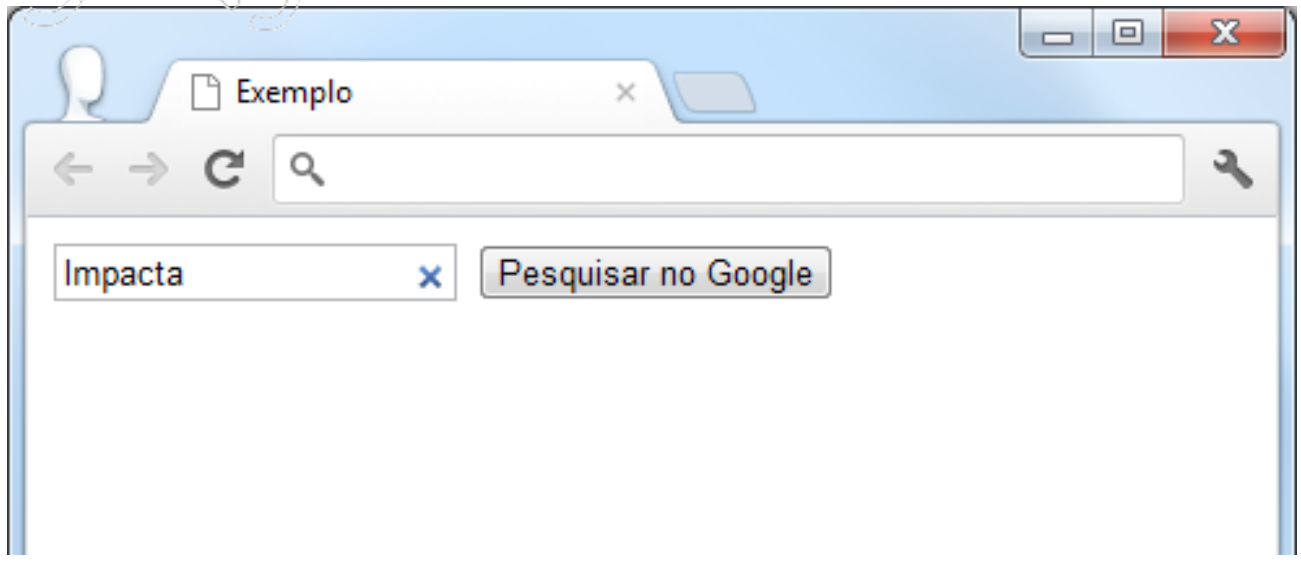
10.4.1.search

Define um controle **input** para busca, diferenciando-o dos outros controles alterando sua estilização e comportamento. Cada navegador possui um modo de estilização e comportamento próprios.

Veja como funciona o atributo **type** com valor **search**:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="https://www.google.com.br/search">
10         <input type="search" name="q" value="Impacta">
11         <input type="submit" value="Pesquisar no Google">
12     </form>
13
14 </body>
15 </html>
```

O resultado é mostrado a seguir:

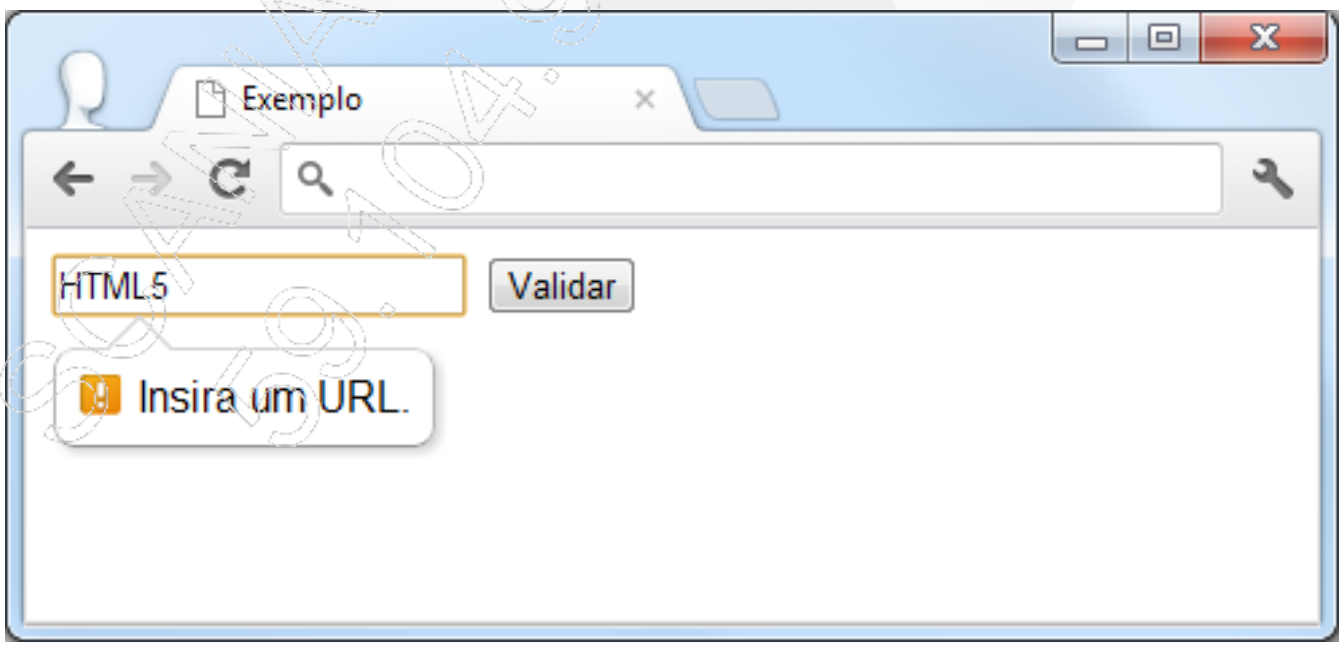


10.4.2.url

Garante que um controle **input** receba uma URL como valor. Ele fornece ao usuário, entre outras coisas, uma indicação visual de que o valor digitado é uma URL válida. Veja como funciona no exemplo a seguir:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="#">
10         <input type="url" name="URL" placeholder="Digite uma URL">
11         <input type="submit" value="Validar">
12     </form>
13
14 </body>
15 </html>
```

Agora veja o resultado:



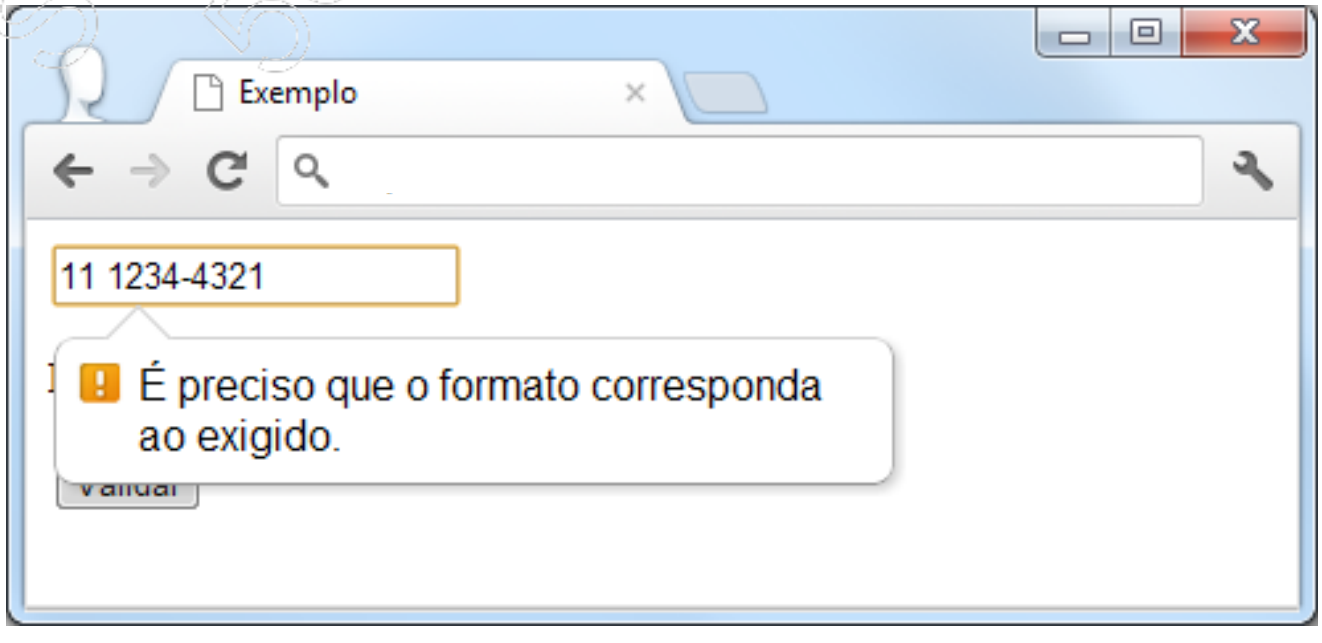
10.4.3.tel

Define um controle **input** destinado a receber um número de telefone. Ele não restringe o controle a uma sintaxe particular porque os formatos de números de telefone podem variar muito. Portanto, ele apenas define o destino do controle.

Para validar um controle desse tipo, você tem duas opções: Usar o atributo **pattern** ou o método **setCustomValidity()**, que é disponibilizado como um mecanismo JavaScript nativo do navegador. O exemplo a seguir mostra como validar esse controle utilizando o atributo **pattern**:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="#">
10         <input
11             type="tel"
12             pattern="\([0-9]{2}\)[\s][0-9]{4}-[0-9]{4}"
13             placeholder="Telefone">
14         <p>Ex: (11) 1234-4321</p>
15         <input type="submit" value="Validar">
16     </form>
17
18 </body>
19 </html>
```

E o resultado você vê logo adiante:



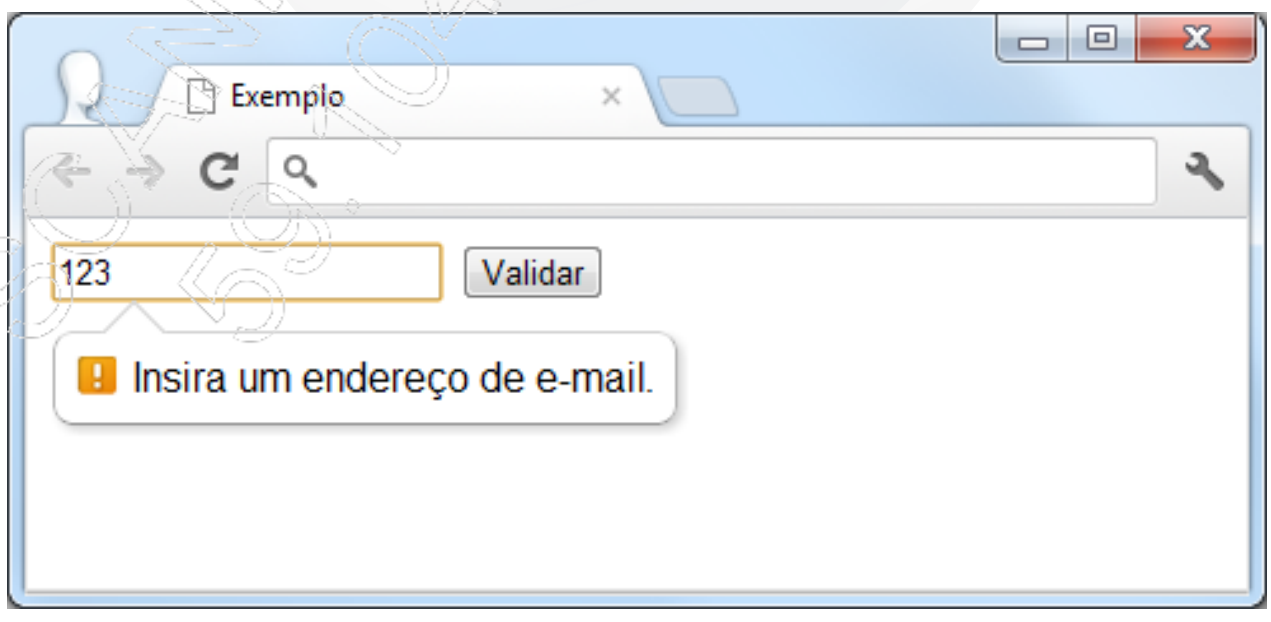
Ao acessar uma página com um input com o **type=”tel”** de um dispositivo móvel, o teclado se adapta para que o usuário possa digitar mais facilmente o valor esperado. Veja:

10.4.4.email

Com este valor, o controle **input** pode receber apenas um endereço de e-mail como valor. Ele, na verdade, não verifica se o e-mail digitado é válido, apenas se o formato digitado é válido. Veja como usá-lo no seguinte exemplo:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="#">
10         <input
11             type="email"
12             placeholder="E-mail">
13         <input type="submit" value="Validar">
14     </form>
15
16 </body>
17 </html>
```

Agora veja o resultado:



Assim como o **type=”tel”**, o **type=”email”** também faz o teclado dos dispositivos móveis se adaptarem para facilitar a entrada de um endereço de e-mail. Veja:

10.4.5.datetime

Um controle **input** com esse valor deverá receber uma string de data e hora UTC. O agente de usuário é quem define como será a representação do grupo data/hora, que dependerá do local e do usuário. É comum o agente de usuário usar um widget do tipo date picker para entradas no controle de data e um campo com slider de seta para entradas de hora.

Veja o uso de **datetime** a seguir:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="#">
10         <input type="datetime" placeholder="Data e Hora">
11         <input type="submit" value="Enviar">
12     </form>
13
14 </body>
15 </html>
```

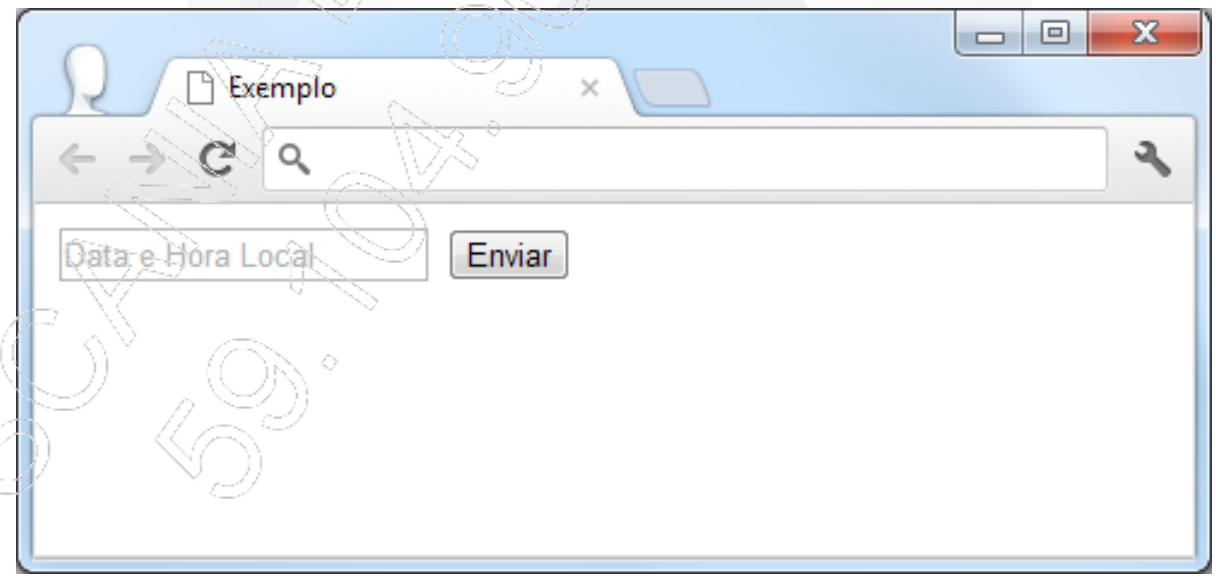
E o resultado é este:

10.4.6.datetime-local

Diferencia-se do **datetime** por aceitar como valor uma string representando data e hora locais, e não UTC. Veja o exemplo a seguir:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="#">
10         <input type="datetime-local" placeholder="Data e Hora Local">
11         <input type="submit" value="Enviar">
12     </form>
13
14 </body>
15 </html>
```

E, em seguida, o resultado:

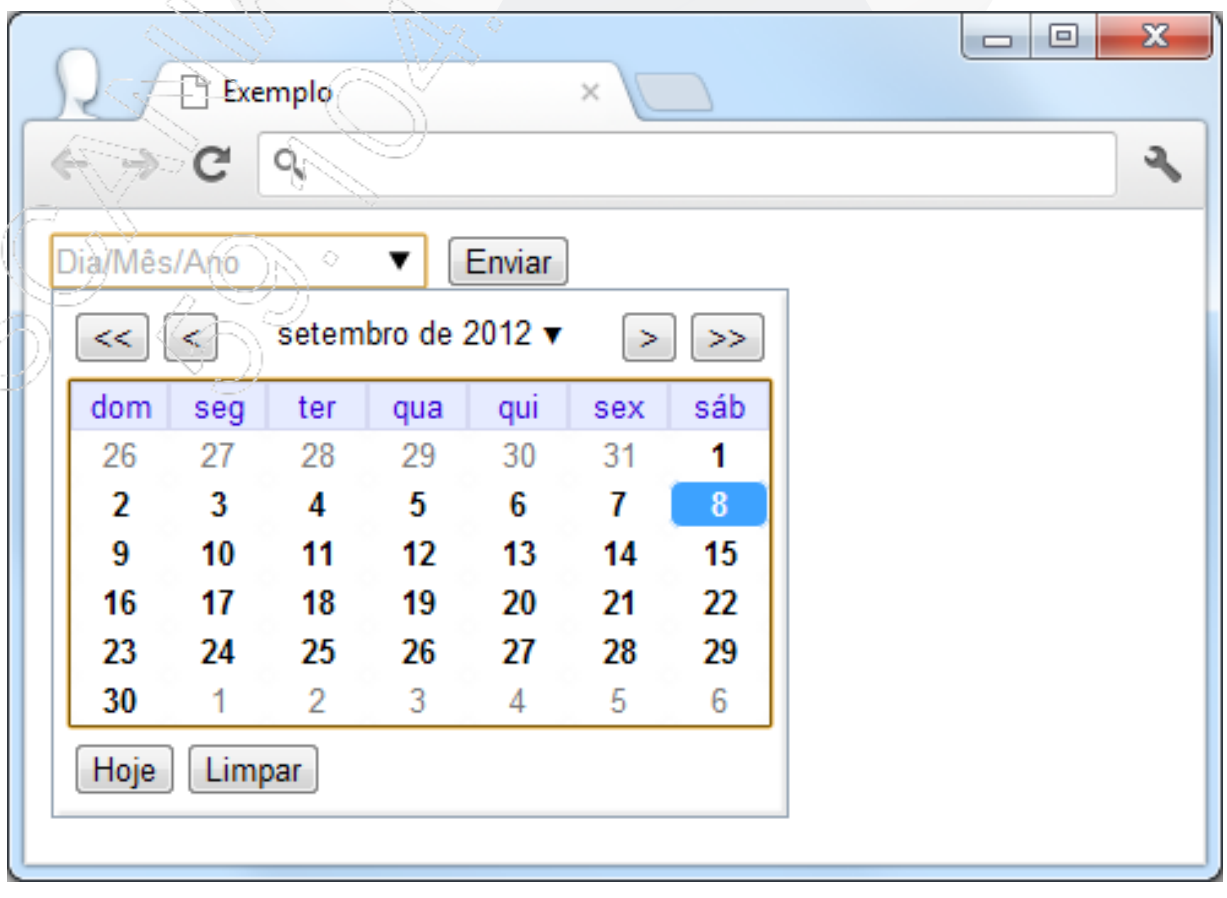


10.4.7.date

Um controle **input** com esse valor deverá receber uma string de data. O agente de usuário é quem define como será a representação da data, que dependerá do local e do usuário. É comum o agente de usuário usar um widget date picker para entradas no controle de data. Veja o exemplo a seguir:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="#">
10         <input type="date" placeholder="Nascimento">
11         <input type="submit" value="Enviar">
12     </form>
13
14 </body>
15 </html>
```

Com o seguinte resultado:



10.4.8.time

Um controle **input** com esse valor deverá receber uma string de hora. Geralmente, o agente de usuário usa um slider de seta para entradas no controle de data. Veja o exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <input type="time" placeholder="Hora">
        <input type="submit" value="Enviar">
    </form>

</body>
</html>
```

10.4.9.month

Define para um controle **input** uma string de mês, cuja representação fica por conta do agente de usuário, de acordo com o local e o usuário. Geralmente, utiliza-se um widget date picker para a entrada no controle. Veja o exemplo a seguir:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <input type="month" placeholder="Mês">
        <input type="submit" value="Enviar">
    </form>

</body>
</html>
```

10.4.10.week

Define um controle que deverá receber uma string representando uma semana do ano. Normalmente, o agente de usuário usa um widget date picker para a entrada. Veja o exemplo a seguir:

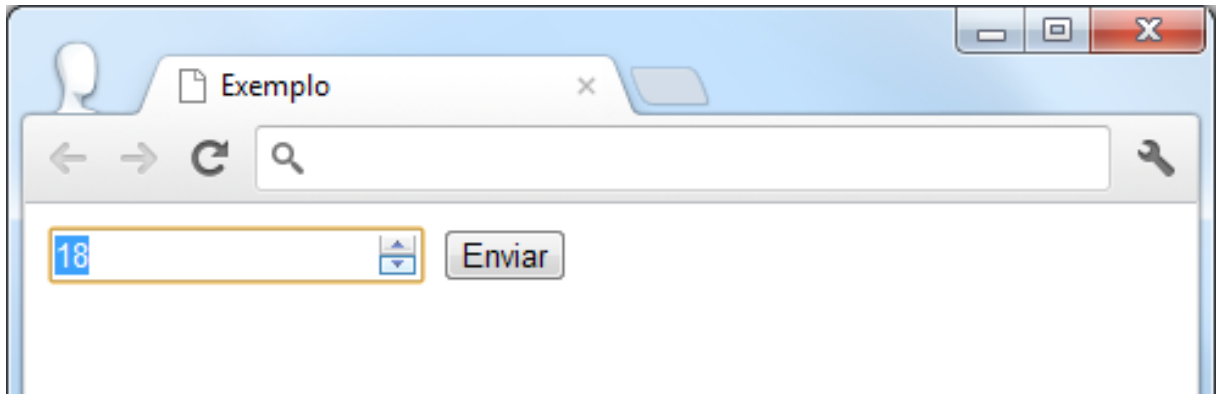
```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <input type="week" placeholder="Semana do Ano">
        <input type="submit" value="Enviar">
    </form>

</body>
</html>
```

10.4.11.number

Controles com esse valor devem receber números. É comum que o agente de usuário use sliders de seta para esse tipo de entrada. Veja um exemplo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <input type="number" min="18" placeholder="Idade">
        <input type="submit" value="Enviar">
    </form>

</body>
</html>
```

O resultado é o seguinte:

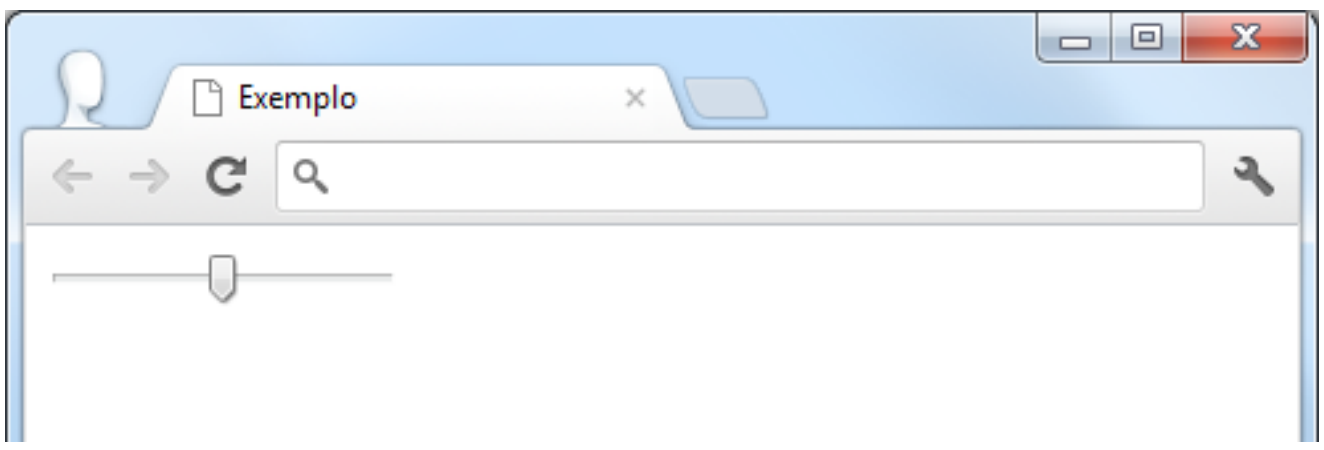


10.4.12.range

Em controles com **range**, a entrada deve ser um número situado dentro de um intervalo definido. É comum que essas entradas tenham um slider deslizante. Veja um exemplo com **range**:

```
1  <!doctype html>
2  <html>
3  <head>
4  <meta charset="utf-8">
5  <title>Exemplo</title>
6  </head>
7  <body>
8
9      <form action="#">
10         <input type="range" min="0" max="100" value="50">
11     </form>
12
13 </body>
14 </html>
```

E o resultado é o seguinte:

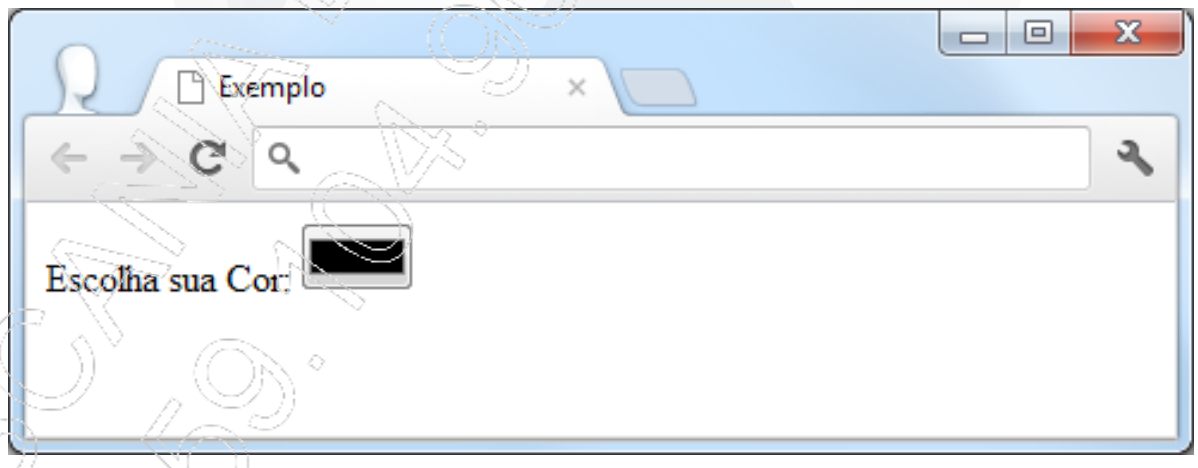


10.4.13.color

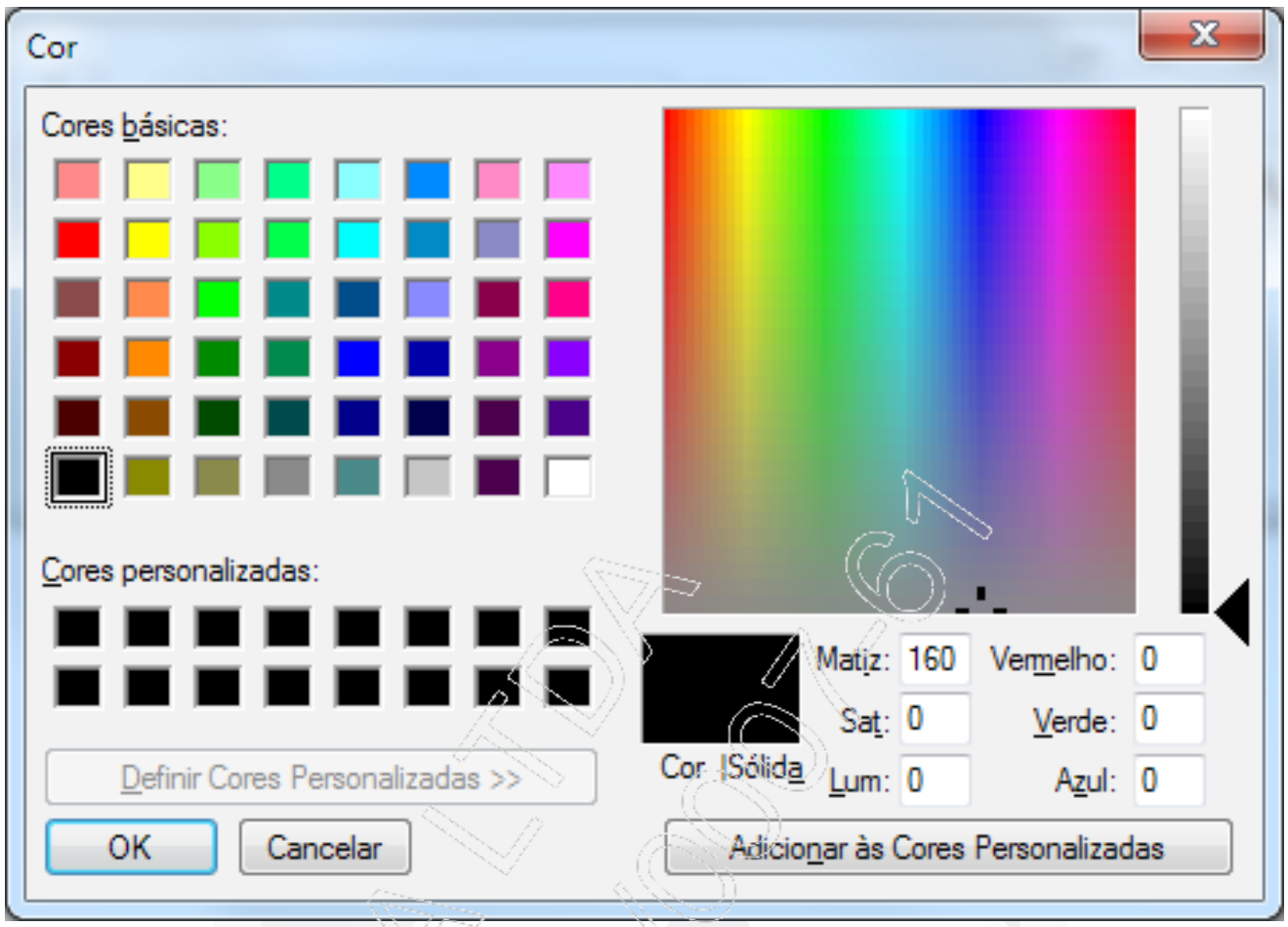
Define que um controle **input** receba um código sRGB de cor. Geralmente, utiliza-se um color pick para esse tipo de entrada. Veja um exemplo:

```
<!doctype html>
<html>
<head>
<meta charset="utf-8">
<title>Exemplo</title>
</head>
<body>

    <form action="#">
        <label for="cor">Escolha sua Cor:</label>
        <input type="color" id="cor" name="cor_escolhida">
    </form>

</body>
</html>
```

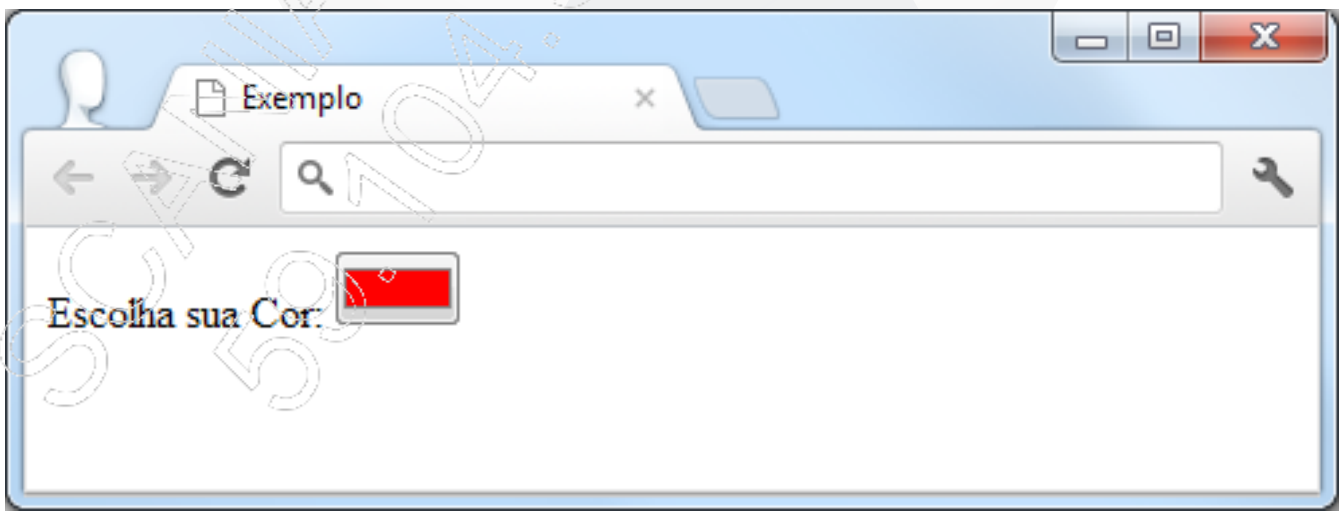
Agora observe o resultado:



Ao clicar no input **type=”color”**, a seguinte janela será aberta:



Após escolher uma cor, o input assume a cor escolhida:



Teste seus conhecimentos

Formulários

10

1. Qual é a seção do documento que abrange texto, marcação, controles e rótulos e tem como função recolher dados do usuário e enviá-los para serem processados no servidor?

- ☐ a) Cabeçalho
- ☐ b) Tabela
- ☐ c) Formulário
- ☐ d) Linguagem
- ☐ e) Rodapé

2. Qual atributo foi adicionado na HTML5?

- ☐ a) form
- ☐ b) name
- ☐ c) value
- ☐ d) rel
- ☐ e) id

3. Qual é o atributo usado para alterar o destino de envio de um formulário?

- ☐ a) formenctype
- ☐ b) formmethod
- ☐ c) formtarget
- ☐ d) formsubmit
- ☐ e) formaction

4. Qual é o atributo usado para validar um campo de formulário com expressão regular?

- ☐ a) regex
- ☐ b) validate
- ☐ c) required
- ☐ d) pattern
- ☐ e) novalidate

5. Qual atributo é usado para dar uma dica de preenchimento de um campo?

- ☐ a) placeholder
- ☐ b) pattern
- ☐ c) dirname
- ☐ d) autofocus
- ☐ e) required

Mãos à obra!

Formulários

10

Laboratório 1

A – Criando um formulário com elementos da HTML5

1. Crie um arquivo HTML com o nome de **formulário.html**;

2. Adicione a estrutura básica de elementos HTML no arquivo;

3. No elemento **BODY**, adicione o elemento **FORM**;

4. Adicione o elemento **LABEL** com o texto **Nascimento:**;

5. Adicione o elemento **INPUT** com o atributo **TYPE** igual a **DATE**;

6. Adicione o elemento **LABEL** com o texto **Salário:**;

7. Adicione o elemento **INPUT** com o atributo **TYPE** igual a **RANGE**, o atributo **MIN** igual a **0** e o atributo **MAX** igual **10000**;

8. Adicione o elemento **LABEL** com o texto **E-mail:**;

9. Adicione o elemento **INPUT** com o atributo **TYPE** igual a **EMAIL**;

10. Adicione o elemento **LABEL** com o texto **Site:**;

11. Adicione o elemento **INPUT** com o atributo **TYPE** igual a **URL**;

12. Adicione o elemento **LABEL** com o texto **Telefone:**;

13. Adicione o elemento **INPUT** com o atributo **TYPE** igual a **TEL** e crie uma expressão regular para validar o valor inserido;

14. Adicione o elemento **BUTTON** com a classe **btn**, o atributo **TYPE** igual a **SUBMIT** e o texto **Enviar**. O resultado final será este:

B – Criando um formulário de autenticação com múltiplos destinos

1. Crie um arquivo HTML com o nome de **login.html**;

2. Adicione a estrutura básica de elementos HTML no arquivo;

3. No elemento **BODY**, adicione o elemento **FORM** com os atributos **ACTION** igual a **aluno.php** e **METHOD** igual a **POST**;

4. Adicione o elemento **INPUT** com o atributo **TYPE** igual a **EMAIL** e o atributo **PLACEHOLDER** igual a **E-mail**;

5. Adicione o elemento **INPUT** com o atributo **TYPE** igual a **PASSWORD** e o atributo **PLACEHOLDER** igual a **Senha**;

6. Adicione o elemento **BUTTON** com o atributo **TYPE** igual a **SUBMIT** e o texto **Login Aluno**;

7. Adicione o elemento **BUTTON** com o atributo **TYPE** igual a **SUBMIT**, o atributo **FORMACTION** igual a **instrutor.php** e o texto **Login Instrutor**. O resultado será como mostrado a seguir:

C – Utilizando campos range para ajustar a largura e a altura de uma imagem

1. Crie um arquivo HTML com o nome de **ajustar_imagem.html**;

2. Adicione a estrutura básica de elementos HTML no arquivo;

3. Adicione os seguintes estilos CSS:

```
body {
    margin:10px;
}
img {
    width:600px;
    height:600px;
}
#Area {
    float:left;
    width:600px;
    height:600px;
    border:rgba(204,204,204,1) 1px solid;
}
#V {
    -webkit-appearance: slider-vertical;
    height:600px;
    width:30px;
    float:left;
}
#H {
    float:left;
    clear:left;
    width:600px;
    height:30px;
}
```

4. Crie um elemento **DIV** com o **ID** igual a **Area** e insira um elemento **IMG** nesse **DIV** com o **SRC** igual a **HTML5-Logo.png**;

5. Crie dois elementos **INPUT** do tipo **RANGE** com os atributos **MAX** e **VALUE** iguais a **600** e **MIN** igual **0**. O primeiro **INPUT** deverá ter o **ID** igual a **V** e o segundo igual a **H**;

6. Crie um elemento **SCRIPT**;

7. Crie uma variável com o nome **ranges** e selecione todos os elementos em que o **TYPE** for igual a **RANGE**;

8. Percorra os elementos selecionados e adicione o evento **change** a cada um deles;

9. No evento, crie uma variável com o nome **imgStyle** que receberá a seleção do atributo **style** do elemento **IMG**;

10. Crie uma nova variável com o nome **size** e defina o valor igual ao atributo **value** do objeto **this** concatenado com a string **px**;

11. Verifique se o **ID** do objeto **this** é igual a **V**. Se a condição for verdadeira, atribua a variável **size** ao atributo **height** da variável **imgStyle**. Senão, atribua ao atributo **width**. O resultado será este: