ELABORATA
INFORMÁTICA

WEBSTANDARDS COM HTML5

Curitiba / 2018

Srs. Alunos

A Elaborata Informática, com o objetivo de continuar prestando-lhe um excelente nível de atendimento e funcionalidade, informa a seguir algumas regras que devem ser observadas quando do uso do laboratório e seus equipamentos, visando mantê-los sempre em um perfeito estado de funcionamento para um melhor aproveitamento de suas aulas.

É proibido:

- Atender celular. Por favor, retire-se da sala, voltando assim que desligar.
- Fazer cópias ilegais de software (piratear), com quaisquer objetivos.
- Retirar da sala de treinamento quaisquer materiais, mesmo que a título de empréstimo.
- Divulgar ou informar produtos ou serviços de outras empresas sem autorização por escrito da direção Elaborata.
- Trazer para a sala de treinamento, qualquer tipo de equipamento pessoal de informática, como por exemplo:
  - Computadores de uso pessoal
  - Notebooks
  - Placas de vídeo
  - Placas de modem
  - Demais periféricos
  - Peças avulsas como memória RAM, ferramentas, etc.
- O consumo de alimentos ou bebidas
- Fumar

Atenciosamente

Elaborata Informática

Sumário

CAPÍTULO 1

INTRODUÇÃO

WEBSTANDARDS COM HTML5

1.1 Introdução à linguagem de marcação

O HTML (Hypertext Markup Language ou Linguagem de Marcação de Hipertexto) é a linguagem com que se formam as páginas da web. Através dela, o navegador (ou browser) traduz (renderiza) a página de uma forma gráfica para que o usuário possa facilmente interagir. O Curso de Web Standards com HTML5 focará a construção de páginas da web através dessa linguagem HTML, a estilização delas através das CSS, e as novidades que novas versões destas linguagens (HTML5 e CSS3) trouxeram.

HTML e CSS são linguagens muito simples e de fácil compreensão, não precisando de um conhecimento pré-requisito em nenhuma outra linguagem de programação para seu entendimento, basta atenção e objetividade. No que concerne as suas novas versões, com efeitos e metodologias de trabalho inovadoras, veremos o porque da necessidade de novas tags e comandos e como isso facilitará a programação web no futuro próximo.

Além de aprendermos na prática o trabalho com as linguagens de marcação, veremos antes de tudo o que são as Web Standards (padrões web), o W3C, a história no que diz respeito a rede mundial atual, e os protocolos de envio de dados.

1.2 A história do HTML

Em 1989, Tim Berners-Lee, físico inglês, procurava um meio de se comunicar com seu grupo de cientistas, o que o levou a criação do conceito de hipertexto, uma forma de comunicar e interligar documentos entre usuários. Em 1990, Tim cria o primeiro navegador capaz de se comunicar entre computadores da NeXT, empresa criada por Steve Jobs em 1985. Comprovado por experiências no CERN (Centro Europeu de Pesquisas Nucleares), foi criado o embrião do que seria hoje a WWW (World Wide Web). Essas informações eram e ainda são transmitidas através de um protocolo HTTP (Hypertext Transfer Protocol), e interpretadas pelo browser.

O conceito foi popularizado em 1992 com a criação do navegador Mosaic pela NCSA (National Center for Supercomputing Applications – Illinois – EUA). Com novos navegadores sendo criados, cada um deles inserindo novas formas de linguagens de marcação (HTML) próprias, a web foi se tornando um caos.

Atualmente a versão HTML em vigor é a 4.01, tendo sua versão 5 em fase de testes. A versão 5 promete além de manter a estrutura atual, trazer novidades

principalmente no que concerne a estruturação do site, inserção de áudio e vídeo e formulários.

1.3 As Web Standards

Para coordenar e padronizar as linguagens de marcação, em 1994, Tim Berners-Lee em parceria com o CERN cria o W3C (World Wide Web Consortium), com sede no MIT (Massachusets Institute of Tecnology), um consócio internacional formado pôr, em média, 300 membros entre empresas, órgãos governamentais, instituições e pesquisadores. Esses padrões são o que chamamos de Web Standards, que nos dizem a forma correta de utilizar as linguagens web até os dias de hoje.

Veja que padronizar de forma alguma busca limitar ou formalizar a criatividade, mas sim tornar as linguagens web possíveis de serem utilizadas por todo o mundo, em diversos navegadores, sem perda de informações e qualidade de navegação. É dever de todos os designers e desenvolvedores web sempre buscar enquadrar seus projetos dentro dos padrões mundiais, para que dessa forma manter, atualizar e administrar os mesmos não se torne uma tarefa árdua e complicada.

1.4 A história das CSS

Atualizar páginas HTML, reestruturar sua aparência, ou manter um site com notícias e novidades constantes era uma tarefa desgastante. Pense em um site de 30 páginas, onde se gostaria de alterar a cor de fundo do mesmo. Sendo feita apenas através da linguagem HTML, precisaríamos fazer a alteração nas páginas uma a uma. Para resolver este problema, em 1996 o W3C adotou sua primeira versão CSS (sigla para Cascading Style Sheets, ou Folhas de Estilo em Cascata). Através dela, propõe-se que nenhuma alteração na aparência do site seja feita por HTML, mas cabe ao HTML apenas dar a estrutura e as informações em si, sem formatação. Toda a estilização do site fica a cargo das CSS, que como veremos, podem ser incorporadas ao código, ou linkadas em arquivos externos para estilização do site todo (estas geralmente mais recomendadas). Esta é uma regra obrigatória quando se adota os padrões web (Web Standards). Desta forma, alterando apenas algumas linhas de código, teremos todo o site com a aparência totalmente reformulada.

Um exemplo prático do poder das CSS é o Zen Garden, um desafio lançado aos Web Designers onde nenhuma alteração no código HTML poderia ser feita, apenas nas folhas de estilo, e assim podemos ver o site se transformando totalmente a cada layout CSS desenvolvido. Confira: http://www.csszengarden.com.

1.5 Semântica

Semântica na língua portuguesa é o estudo do significado das palavras. Na *web,* chamamos de semântica a utilização das *tags* e comandos para aquilo que realmente servem. É possível, mas não é semântico, utilizar comandos para uma função, sendo que na verdade ela foi criada para outra.

1.6 Acessibilidade

O bom profissional *web* desenvolve seus projetos levando em conta a acessibilidade. Acessibilidade é a preocupação com que o projeto possa ser acessado por qualquer tipo de usuário, desde usuários usuais até usuários portadores de necessidades especiais. Para isso, é necessário que o projeto traga alguns comandos que possibilitem o acesso a estes usuários.

Também entra no fator acessibilidade o trabalho para que o projeto tenha a possibilidade de ser exibido em diversos tipos de navegadores diferentes, dispositivos e resoluções.

1.7 O protocolo HTTP

É necessário para um bom Web Designer entender como funciona o envio e recebimento das informações pela web, para assim poder trabalhar melhor com imagens, códigos e facilitar o entendimento da página. O Protocolo HTTP (Hypertext Tranfer Protocol, ou Protocolo de Transferência de Hipertexto) é a forma que se dá a comunicação entre o cliente (navegador) e um servidor web, onde o site está hospedado. Este protocolo é usado pela World Wide Web desde 1990. Essa comunicação se dá no padrão requisição-resposta, ou seja, o cliente envia uma requisição de um recurso, e o servidor encaminha a resposta em forma de pacotes de dados, presumindo que a conexão estabelecida tenha tido sucesso. Assim, quando o cliente pede um documento HTML, o servidor envia o documento ao usuário juntamente

com todos os pacotes de dados a ele ligados, sejam eles imagens, vídeos, animações, ou arquivos de programação a ele linkados, como CSS, PHP ou Java Script.

1.8 O que é XHTML

O XHTML é a reformulação do código HTML, com os conceitos e regras do XML (Extensible Markup Language). O XML é mais rígido em sua sintaxe, o que a torna uma linguagem universal. Os documentos XHTML serão exibidos no navegador da mesma forma como o HTML, só que mais compatível aos navegadores existentes. Por exemplo, na prática poderíamos criar uma página com regras simples, sem descrever o documento em que estamos trabalhando, mas isto funcionaria na prática? Talvez sim, mas de forma desordenada, em poucos navegadores e com fortes tendências a não se manter em demais idiomas e padrões de caracteres. Já em um documento XHTML, devemos descrever o documento, dizer a linguagem utilizada, o tipo de formatação, isto só para iniciarmos nosso projeto.


ELABORATA
INFORMÁTICA
CAPÍTULO 2
ENTENDENDO O XHTML

WEBSTANDARDS COM HTML5

2.1 Entendendo o XHTML

As páginas *web* (sites ou sítios) são sempre hospedadas em um servidor, e acessadas pelo cliente através de um endereçamento. Esse endereçamento é no formato numérico (**IP** ou ***Internet Protocol***), que pode ser visualizado através do *prompt* de comando do *Windows* através do comando ping "nome do site". Para facilitar o acesso por parte dos usuários, registra-se um domínio com um nome para que o mesmo seja acessado através de uma **URL**. Quando o cliente faz uma solicitação através de uma **URL**, o pacote de dados retornado poderá ser um arquivo no formato **XHTML**, que é formado por *tags* e comandos que estruturam e formam o conteúdo de toda a página. Esses comandos são traduzidos pelo navegador e exibidos ao usuário no formato de imagens, textos e objetos inseridos na página. Podemos visualizar o código **XHTML** de qualquer site da *web* através da opção no navegador **Exibir código-fonte**, em alguns casos, **CTRL+U**.

Existem outras formas de se escrever o código **HTML** que talvez funcionariam na prática, mas estariam fora dos padrões do **W3C**. No nosso caso, aprenderemos apenas a forma correta de escrever o código, dentro dos padrões internacionais.

2.2 Sintaxe da linguagem

A estruturação do **XHTML** é baseado em *tags*, ou etiquetas, que são comandos que dizem ao navegador o tipo de formatação ou estrutura que deve ser criada na página, como tabelas, parágrafos, cabeçalhos e uma infinidade de outras funções. Toda *tag* é iniciada por um sinal de menor (**<**) e encerrada por um sinal de maior (**>**). Entre esses sinais teremos a descrição da *tag* em si, como por exemplo, a *tag* ** ** para parágrafos, ou **<img />** para imagens. Ao inserir uma *tag* o navegador a interpretará como um comando, e insere o elemento desejado.

Também existe a *tag* de fechamento, que contém uma barra (*/*), como em ** **. Desta forma o texto ou objeto que sofre interferência ficará entre a *tag* de abertura e a *tag* de fechamento, como em: ** **Elaborata Informática** **. No exemplo, o texto está contido em um parágrafo.

Em alguns casos podem existir *tags* que não necessitam de fechamento, mas fecham em si mesmas, por nunca haver a necessidade de colocar um elemento dentro

dela, como no caso de imagens (**<img />**), quebras de linha (**
), linha horizontal (<hr />**), entre outros. Neste caso, a barra é inserida no final da própria *tag* de abertura.

As *tags* que possuem *tag* de fechamento são chamados de **Elemento Conteúdo**, e as que fecham em si mesmas de *tag* de **Elemento Vazio**. Veja o exemplo:

Tag de elemento conteúdo	Tag de elemento vazio
Texto em parágrafo	<hr />
Texto em div	
Texto em negrito	<input type=”button” value=”enviar” />

2.3 Atributos

Algumas *tags* precisam de mais especificações para chegarmos ao objetivo desejado. Essas especificações serão descritas dentro da própria *tag* e são chamadas de atributos. Como por exemplo, a *tag* **<table>** iniciará uma tabela, mas sem bordas, cores, identificação e espaçamentos. Se quisermos dar propriedades a esta tabela, como bordas, colocaríamos o atributo *border* dentro da *tag* **<table>** para fazer a solicitação, como no exemplo:

```
<table border=”1”>
```

Note que o atributo é colocado logo depois da *tag,* mas ainda antes do sinal de maior, separado com um espaço. Note também que todo o atributo precisa de uma especificação do seu respectivo valor (no caso, largura da borda da tabela igual a 1 pixel), e que o valor vem sempre depois de um sinal de igual e fica entre aspas duplas. Essa será a sintaxe padrão para todos os atributos de *tags*.

Existem atributos que são chamados de atributos globais, ou seja, podem ser inseridos em qualquer *tag* do documento. Esta lista de atributos globais está ainda maior agora com o **HTML5**. Já grande parte dos atributos só funcionam se inseridos em determinadas *tags* específicas. Vamos conhecer algumas regras para o uso correto do **XHTML:**

- As *tags* devem ser escritas em letras minúsculas;

A metalinguagem **XML** é *case-sensitive* (sensível a maiúsculas e minúsculas), assim como o **XHTML**. Sempre que formos escrever os códigos utilizaremos somente letras minúsculas.

- As *tags* devem ser aninhadas;

Ao se abrir duas *tags* sequênciais, como no caso de precisar colocar um texto em negrito de um parágrafo que está dentro de uma divisão, a formatação correta seria a seguinte:

```
<div><p><strong>texto em negrito</strong></p></div>
```

Note que a primeira *tag* aberta foi a última a ser fechada, e a última *tag* aberta foi a primeira a ser fechada.

- Sempre fechar as *tags*;

Nunca se esqueça de informar na página **XHTML** em qual momento determinada *tag* deve parar de agir, fechando-a. No caso de *tags* de **Elemento Vazio**, nunca se esqueça de colocar a barra de fechamento ao final dela.

- Todo o atributo deve possuir o sinal de igual, e o valor dele colocado entre aspas.

Ao definir o valor do atributo de determinada *tag*, sempre deve haver um sinal de igual e seu valor entre aspas duplas:

```
<tag atributo=”valor”>formatação</tag>
```

Há raras exceções em que o atributo não tem valor, por não haver necessidade de especificação, esses chamados de atributos *booleanos*, como em:

```
<input type=”checkbox” name=”cursos” checked />
```

2.4 Estrutura básica

Vamos para a prática. O código **XHTML** pode ser criado em um software de texto qualquer, desde o ***Dreamweaver***, uma **IDE** para o trabalho para códigos *web*, até o **Bloco de Notas**. No nosso caso utilizaremos um software gratuito chamado ***Notepad++***, muito semelhante ao **Bloco de Notas** do *Windows*, mas com algumas funções a mais que nos ajudarão a entender melhor o nosso código. Depois de abrir o *software*, teremos algumas *tags* padrões a serem inseridas, que devem existir em todo o documento **XHTML** para podermos começar, são elas:

- **DTD** (*Document Type Definition*);

É a definição do tipo de documento, que diz ao navegador o tipo de site que estamos criando. Ele deve ser escrito na primeira linha do código, não podendo haver nada antes dele. Podem ter três variações:

Strict:

Este **Doctype** é utilizado em casos de documentos com a **XHTML** mais restrita, que não aceita comandos que não sigam as normas e padrões do **W3C**.

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

Transitional:

Doctype mais comumente utilizado. Tem as restrições de um **XHTML** comum, mas aceita algumas variações de sintaxe. Também utilizado para o caso de dispositivos que não possuam suporte as **CSS**.

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

Frameset:

Este tipo de documento é utilizado somente se a página possuir *frames*, efeito em desuso nos dias atuais.

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

Dica: *Observe que o **DTD** é um código grande e de difícil assimilação. Esta será a primeira inovação do **HTML5** que veremos. No **HTML5** a **DTD** ficou como <!DOCTYPE html>.*

- *Tag* **<html>**

Tag que dá início ao projeto. Ela é fechada ao final do projeto com **</html>**. Nos padrões comuns do **W3C**, dá-se um atribulo a ela que indica as específicas regras do seu **XML**, que é o atributo **xmlns** e usa como valor um *link* onde essas regras se encontram, ficando assim:

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

- *Tag* **<head>**

Define o cabeçalho da página. O que inserirmos entre a *tag* **<head>** e a **</head>** não entrará na página como elemento físico, mas irá influenciá-la, como no caso da *tag* **<title>** que veremos a seguir, *meta-tags* e *tags* de ligação com documentos externos.

- *Tag* **<title>**

O *title* ou título do site é o nome que aparecerá na barra de título do navegador, em suas guias (se houver), e na barra de tarefas do *Windows* (no caso de versões mais antigas). Permanece entre as *tags* **<head>** e **</head>**. Colocaremos o título da página entre as *tags* de abertura e fechamento:

```
<title>Título da página</title>
```

- *Tag* **<body>**

O corpo do site, onde serão inseridas as informações em si. Ao final, a estrutura básica para iniciarmos qualquer projeto ficará assim:

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

<html xmlns="http://www.w3.org/1999/xhtml">
      <head>
            <meta   http-equiv="Content-Type"   content="text/html;charset=utf-
8" />
            <title>Elaborata Informática</title>
      </head>
      <body>

      </body>
</html>
```

Nota: *A meta tag charset diz ao navegador o tipo de caracteres que está sendo utilizado na página. O valor utf-8 é o padrão atual para nosso idioma. Você pode substituir o valor utf-8 por outro valor, como iso-8859-1 para latin-1, mas ele garante menos suporte para os caracteres atuais.*

No caso do documento utilizando as inovações do **HTML5** ficará assim:

```
<!DOCTYPE html>

<html lang="pt-br">
      <head>
            <meta charset="UTF-8" />
            <title>Elaborata Informática</title>
```

```
    </head>
    <body>


    </body>
</html>
```

O **HTML5** encurtou o **DTD,** facilitou o entendimento da linguagem utilizada no documento e o tipo de caracteres.

	***Obs.:*** *Note que os códigos foram organizados com tabulações para facilitar seu entendimento, chamamos isso de indentação. Não interfere na função do código, mas facilitará a leitura e atualização do mesmo.*

CAPÍTULO 3

CRIANDO UM DOCUMENTO XHTML

WEBSTANDARDS COM HTML5

3.1 Criando um documento XHTML

Após digitarmos toda a estrutura básica do **XHTML** no software adequado, devemos salvá-la corretamente. Para isso, acesse menu ***Arquivo/Salvar como...*** e defina o nome do arquivo seguido da extensão .html. Caso esteja criando a página inicial do site, salve com o nome **index.html** para que o servidor identifique o documento como página inicial. Podemos abrir este arquivo normalmente para visualizá-lo no navegador desejado, ou clicar com o botão direito sobre ele, escolhendo a opção ***Edit with Notepad++*** para continuar o código.

Obs.: *Os nomes de todos os arquivos do projeto que serão publicados na web não podem conter espaços em branco, acentos, cedilhas ou caracteres especiais.*

Dica: *Crie uma pasta para armazenar todos os arquivos do seu projeto, com subpastas para cada categoria de arquivos, como imagens, animações, CSS, PHP e assim por diante.*

3.2 Formatação de texto

Atualmente, a formatação de textos é feita quase toda por **CSS.** Mas é necessário a inserção de uma *tag* no documento para que as **CSS** possam estilizá-lo. Veremos como fazer essa formatação por **XHTML**, e mais tarde sua estilização pelas folhas de estilo. As principais *tags* para formatação de textos são:

Tag	Função
<u></u>	Sublinhado
ou	Negrito
ou	Itálico
	Pode ter os atributos *face* para tipo da letra, *size* para tamanho e *color* para cor.
	Novo parágrafo. Pode conter atributos como *align* para alinhamento do texto
 	Quebra de linha
, , , ...	*Tags* de Cabeçalho

Veja o exemplo a seguir:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Formatação de textos</title>
      </head>
      <body>
            <p align=”justify”><strong>Web Standards</strong> é um conjunto de <u>normas, diretrizes, recomendações, notas, artigos, tutoriais e afins</u> de caráter técnico, produzidos pelo <font color=”#009933”><i>W3C</i></font> e destinados a orientar fabricantes, desenvolvedores e projetistas. <br /> Seu uso possibilita a criação de uma <em>web</em> acessível a todos, independentemente dos dispositivos usados ou de suas necessidades especiais.</p>
      </body>
</html>
```

A **tag <font>** pode possuir atributos como *face* (tipo da letra), *size* (tamanho da letra) e *color* (cor da letra). Ao utilizar o atributo *face*, utilize como valor do atributo apenas fontes padrão, ou seja, fontes existentes em todos os sistemas atuais. Ao se utilizar o atributo *size*, utilize variações de valor entre -2 e 4 para escolha do tamanho. Ao utilizar o atributo *color*, utilize o nome da cor em inglês ou seu respectivo código de cores hexadecimal. Não abordaremos detalhadamente esta formatação pois está em desuso, veremos aprofundadamente ao estudarmos as **CSS**, padrão atual.

A *tag* **<p>** possui entre outros atributos, o atributo *align* para alinhamento de textos, que pode variar em valores como *left* (esquerda), *center* (centralizado), rig*h*t (direita) ou *justify* (justificado).

3.2.1 Listas

O **XHTML** possui três tipos de listas:

- Lista não ordenada, definida pela *tag* **
**;
- Lista ordenada, definida pela *tag* **
**;
- Lista de definição, definida pela *tag* **<dl></dl>.**

No caso das listas ordenadas e não ordenadas, cada item da lista será definido pela *tag* **
. No caso da lista de definição, cada item da lista será definido pelas *tags* **<dt></dt> e **<dd></dd>**. Exemplo de lista ordenada:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Lista ordenada</title>
      </head>
      <body>
            <ol>
                  <li>Item 1</li>
                  <li>Item 2</li>
                  <li>Item 3</li>
            </ol>
      </body>
</html>
```

Obs.: *A lista ordenada pode conter o atributo type dentro da tag <ol> com a variação de valores A, a, i, I ou 1.*

Exemplo de lista não ordenada com sub-itens:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Lista não ordenada</title>
      </head>
      <body>
            <ul>
                  <li>Webdesigner</li>
                        <ul>
                              <li>Dreamweaver</li>
```

```
                        <li>Fireworks</li>
                        <li>Flash</li>
                        <li>Flash com Action Script</li>
                    </ul>
                <li>Designer Gráfico</li>
                    <ul>
                        <li>Indesign</li>
                        <li>Photoshop</li>
                        <li>CorelDRAW</li>
                    </ul>
            </ul>
        </body>
</html>
```

Obs.: *A lista não ordenada pode conter o atributo type dentro da tag <ul> com a variação de valores square, circle ou disc.*

Exemplo de lista de definição:

```
<!DOCTYPE html>
<html lang=”pt-br”>
    <head>
        <meta charset=”UTF-8” />
        <title>Lista de definição</title>
    </head>
    <body>
        <dl>
            <dt>Lista de Definição 1</dt>
            <dd>Item 1</dd>
            <dd>Item 2</dd>
            <dt>Lista de Definição 2</dt>
            <dd>Item 1</dd>
            <dd>Item 2</dd>
```

```
            </dl>
      </body>
</html>
```

3.2.2 Cabeçalhos

As *tags* de cabeçalho (ou *headings)* são extremamente difundidas e utilizadas nos dias de hoje. Elas são seis: **<h1></h1>**, **<h2></h2>**, **<h3></h3>**, **<h4></h4>**, **<h5></h5>** e **<h6></h6>**. Se utilizadas apenas com o padrão **XHTML**, elas alteram o tamanho da letra, sempre em negrito e fazem a quebra de linha automaticamente. Mas sua maior utilidade hoje será estilizar estes cabeçalhos por **CSS**, para que toda notícia, título, subtítulo e assim por diante de todo o site tenha a mesma formatação. Veja o exemplo:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Cabeçalhos</title>
      </head>
      <body>
            <h1>Cabeçalho 1</h1>
            <h2>Cabeçalho 2</h2>
            <h3>Cabeçalho 3</h3>
            <h4>Cabeçalho 4</h4>
            <h5>Cabeçalho 5</h5>
            <h6>Cabeçalho 6</h6>
      </body>
</html>
```

3.3 Inserindo elementos

3.3.1 Linha horizontal

Usualmente utiliza-se uma linha horizontal para separar notícias e conteúdos, podemos inseri-las através da tag de elemento vazio **<hr />**. Ela pode ser estilizada com o uso de atributos como *align* para alinhamento e *width* para largura.

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Linha horizontal</title>
      </head>
      <body>
            <center>
                  <h1>Cabeçalho 1</h1>
                  <hr width=”300” />
                  <h2>Cabeçalho 2</h2>
            </center>
      </body>
</html>
```

Obs.: *Podemos utilizar a tag para centralizar qualquer elemento.*

3.3.2 Imagens

Para inserir imagens em um documento, devemos antes tomar algumas precauções:

- Você deve ter o direito de uso sobre a imagem antes de colocá-la no seu projeto. Caso prefira, existem alguns bancos de imagens gratuitos na internet;
- Baixe e insira todas as imagens em uma subpasta da pasta do seu projeto, para organizar e facilitar o *upload* a página depois;

- Preocupe-se com o tamanho do arquivo em pixels, para ter imagens em boa resolução, mas procure equilibrar isso com o tamanho da imagem em *bytes*, para não "pesar" muito o seu projeto.

Assim como as *tags* de formatação de texto, a *tag* para inserir imagens deve ser colocada na seção **<body></body>** da estrutura do código **XHTML**. A *tag* para inserir imagens é a *tag* **<img />**, que é uma *tag* de elemento vazio. Este comando sozinho não fará nenhuma alteração na página, precisamos dar alguns atributos para inserirmos a imagem corretamente. São eles:

Atributo da tag <img />	Função
src	Determina o caminho para a imagem que gostaríamos de inserir
alt	Texto alternativo para que deficientes visuais possam saber qual objeto está inserido
title	Texto que aparecerá quando o usuário parar o mouse sobre a imagem
border	Define uma borda para a imagem. Valor em pixels
width	Largura da imagem. Valor em pixels
height	Altura da imagem. Valor em pixels
align	Alinhamento da imagem. Pode ter como valores *top, middle*, *bottom*, *left* e *right*, influenciando seu alinhamento e a interação da imagem em relação ao texto

Veja o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
      <head>
            <meta charset="UTF-8" />
            <title>Inserindo imagens</title>
      </head>
      <body>
            <img            src="http://elaborata.net.br/webstandards/logo.jpg"
title="Logo da Elaborata" alt="Logo da Elaborata" border="1" width="200"
height="150" align="left" />
      </body>
</html>
```

Ao definir *link* na imagem, sempre defina o atributo *border* com o valor 0 (Zero) por questão estética. Sempre insira a extensão da imagem, que geralmente varia entre .jpg, .gif ou .png.

	***Dica:*** *O atributo title é um atributo global. Pode ser inserido em qualquer tag para que o texto contido em seu valor apareça quando o usuário posicionar o ponteiro do mouse sobre o elemento.*

3.4 Comentários

Para uma melhor compreensão do código inserido, é de bom grado que se comente as regiões do código correspondentes a cada conteúdo. Para comentar o código **XHTML**, faremos da seguinte forma:

```
<!-- aqui inserimos o comentário -->
```

Podemos comentar qualquer área do código, não prejudicando em nada sua sintaxe ou renderização, pois o navegador sempre irá ignorá-lo.

CAPÍTULO 4

META TAGS

4.1 Entendendo as meta tags

As **meta tags** são *tags* inseridas na seção **<head></head>** da página, e tem o intuito não de inserir um conteúdo propriamente dito na página, mas sim inserir valores como o autor da mesma, as palavras-chave para busca em sites especializados ou a descrição da mesma. As *tags* **<meta />** são de elemento vazio, fechando nelas mesmas. Contém os atributos **name**, para especificação do tipo de **meta tag**, **http-equiv**, semelhante a **name** para controle **http** e **content**, que informa o conteúdo da **meta tag**. Conheça as principais:

- *Description* (descrição):

```
<meta name="description" content="Referência em formação profissional. Mais de 90 cursos na área de TI" />
```

Informa o texto de descrição que aparecerá logo abaixo de um resultado em um dispositivo de busca.

- *Keywords* (Palavras-chave):

```
<meta name="keywords" content="xhtml, css, webstandards, cursos, TI, informática, informatica, html5" />
```

Informa palavras-chave que funcionarão como referência para os dispositivos de busca. Separam-se os termos por vírgula.

- *Author* (Autor):

```
<meta name="author" content="Elaborata Informática" />
```

Informa o profissional ou empresa de autoria do projeto.

- *Copyright* (Direitos autorais):

```
<meta name="copyright" content="2007 Elaborata Informática" />
```

Define os parâmetros de direitos autorais da página ou conteúdo.

- *Refresh* (atualizar):

```
<meta http-equiv="refresh" content="3" />
<meta http-equiv="refresh" content="3;URL=http://www.elaborata.com.br" />
```

No primeiro exemplo, temos a atualização automática de uma página a cada três segundos, definido no valor do atributo **content**. Já no segundo temos o

redirecionamento da página para o endereço definido após 3 segundos, utilizado por exemplo em páginas de confirmação de envio de informações.

- *Rating* (Classificação):

```
<meta name="rating" content="general" />
```

Define uma classificação etária para o site. Os valores do atributo **content** podem variar entre *general,* para qualquer idade, *14 years*, por exemplo, para censura de 14 anos ou *mature,* para maiores de 18 anos. Alguns navegadores utilizam esta **meta tag** como parâmetro para filtro.

- *Generator* (Gerador):

```
<meta name="generator" content="Notepad++" />
```

Serve apenas para informação. Informa o tipo de *software* utilizado para a criação do código-fonte.

- *Language* (Idioma):

```
<meta http-equiv="content-language" content="pt-br" />
```

Utilizado em documentos **HTML 4.01** para definição do idioma utilizado. Padrão **W3C** para esse tipo de documento. Como vimos, no **HTML5** definiremos o idioma com o atributo *lang* na *tag* **<html></html>**.

- *Charset* (Tipo de caracteres):

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

Também utilizado em documentos **HTML 4.01**, mas para definição do tipo de caracteres utilizado. Padrão **W3C** para esse tipo de documento.

- *Charset* (Tipo de caracteres) em **HTML5**:

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

No **HTML5** utilizaremos a **meta tag charset** para a definição do tipo de caracteres. Padrão **W3C** para esse tipo de documento. É necessário sempre definir o tipo de caracteres no documento para que o navegador possa interpretar os caracteres utilizados. Por padrão utiliza-se o padrão utf-8, mas podem haver variações, como a Latin-1. Caso não seja definido, deve-se utilizar um código que gere o caracter desejado, confira a tabela completa desses caracteres no capítulo **Tabelas de referência**.

Exemplo de aplicação das **meta-tags**:

```
<!DOCTYPE html>
<html lang="pt-br">
      <head>
            <meta charset="UTF-8" />
            <title>Aprendendo meta-tags</title>
            <meta name="description" content="Mais de 17 anos em cursos profissionalizantes" />
            <meta name="keywords" content="xhtml, css, webstandards, cursos, TI, informática, informatica, html5, css3" />
            <meta name="author" content="Diego" />
            <meta name="copyright" content="2007 Elaborata Informática" />
            <meta name="rating" content="general" />
            <meta name="generator" content="Notepad++" />
      </head>
      <body>


      </body>
</html>
```

Obs.: *Você pode inserir a quantidade de meta-tags que achar necessário, desde que siga o padrão W3C de definir idioma e tipo de caracteres. As meta-tags podem ser inseridas em qualquer momento da seção <head></head> do documento.*

4.2 Elemento Link

Outra *tag* comumente inserida na seção **<head></head>** da página é a **<link />**. A *tag* **<link />** ligará o arquivo onde ela é inserida a um outro arquivo externo de comportamento ou estilização que influenciará o documento **XHTML**. Por exemplo, podemos linkar um documento em uma folha de estilos **CSS** que irá estilizá-lo. Veja:

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

```
            <title>Elemento Link</title>
            <link rel=”stylesheet href=”estilos.css” />
      </head>
      <body>

      </body>
</html>
```

No exemplo foram definidos atributos para indicar o tipo de ligação feita e a localização do arquivo. Veremos com mais detalhes no capítulo de **CSS**.


ELABORATA
INFORMÁTICA
CAPÍTULO 5
ESTRUTURANDO O LAYOUT

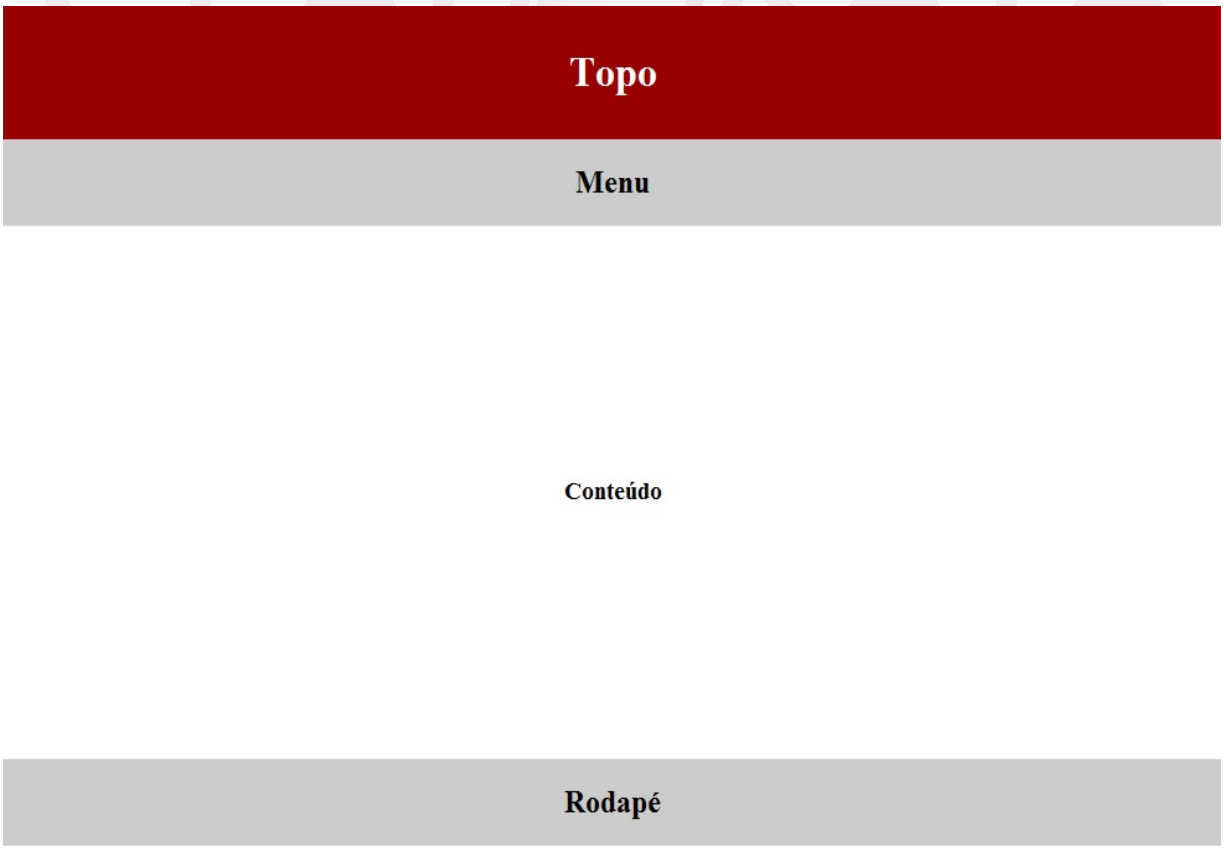
WEBSTANDARDS COM HTML5

5.1 Estruturando o layout

Layout é uma palavra que significa formato. No caso das páginas **XHTML** significará a forma em que o site está estruturado. Você já deve ter percebido que escrever o texto e estilizá-lo por **XHTML** não tem segredo, mas organizá-lo de forma a ter a aparência de uma página *web* é o que faremos agora. Este processo pode ser feito de várias formas: Através de tabelas, de **div's**, **frames**, **iframes** ou pelas novas *tags* **HTML5.** Vamos começar estudando o *layout* feito em tabelas.

5.2 Layout por tabelas

Tabelas por padrão são utilizadas para a organização de dados tabulares, mas por muito tempo foram utilizadas para estruturação de *layouts*. Tabelas são a forma mais primitiva de se construir um *layout*. Por mais que nos dias atuais estejam entrando em desuso, ainda hoje são utilizadas para estruturação de *mail marketing* ou páginas sem suporte as **CSS**. Combinando células, dividindo-as e formatando-as podemos construir todo o *layout* de uma página, como no exemplo:

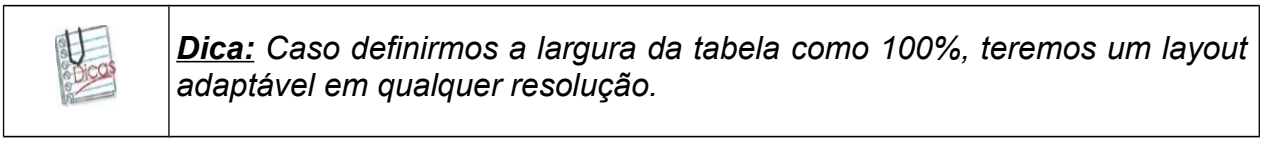


O código **XHTML** que gerará esta página é o seguinte:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Layout por tabelas</title>
      </head>
      <body>
            <center>
                  <table      summary=”layout”      width="984"      border="0"
cellspacing="0" cellpadding="0">
                        <!-- Aqui começa o topo -->
                        <tr>
                              <td          height="100"          align="center"
bgcolor="#990000"><h1>Topo</h1></td>
                        </tr>
                        <!-- Aqui termina o topo -->
                        <!-- Aqui começa o menu -->
                        <tr>
                              <td          height="50"          align="center"
bgcolor="#CCCCCC"><h2>Menu</h2></td>
                        </tr>
                        <!-- Aqui termina o menu -->
                        <!-- Aqui começa o conteúdo -->
                        <tr>
                              <td                                    height="400"
align="center"><h3>Conteúdo</h3></td>
                        </tr>
                        <!-- Aqui termina o conteúdo -->
                        <!-- Aqui começa o rodapé -->
                        <tr>
                              <td          height="50"          align="center"
bgcolor="#CCCCCC"><h2>Rodapé</h2></td>
                        </tr>
                        <!-- Aqui termina o rodapé -->
```

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

Note que existe apenas uma tabela de quatro linhas, e cada linha representando uma seção da página. Poderíamos também, na área conteúdo por exemplo, inserir mais uma tabela dentro desta célula para estruturarmos o conteúdo de cada página. O código está comentado para um melhor entendimento de cada área, com o código ainda pequeno pode não parecer necessário, mas esta é apenas a estrutura, tudo o mais será inserido nela, e estes comentários podem vir a calhar quando o código estiver bem maior.



Dica: *Caso definirmos a largura da tabela como 100%, teremos um layout adaptável em qualquer resolução.*

Como podemos observar, a *tag* para a criação de tabelas é a **<table></table>**. Mas ela de forma isolada, apenas inicia e finaliza a região para a tabela, ainda é preciso dar-lhe propriedades e criar as células. Os atributos passiveis de serem inseridos na *tag* **<table>** são:

Atributo da tag <table></table>	Função
border	Insere a borda da tabela. Valor em pixels
width	Largura da tabela. Valor em pixels
height	Altura da tabela. Valor em pixels
cellspacing	Espaçamento entre células. Valor em pixels
cellpadding	Espaçamento entre o início da célula e o conteúdo inserido. Valor em pixels
align	Alinhamento da tabela. Pode ter valores como *center, left* ou *right*
bordercolor	Cor da borda. O valor pode ser o código hexadecimal da cor ou seu nome em inglês
bgcolor	Cor de fundo. O valor pode ser o código hexadecimal da cor ou seu nome em inglês
background	Imagem de fundo. O valor será o caminho para imagem, seguido de seu nome e extensão
summary	Descrição da tabela

Obs.: *Para fazermos uma tabela "invisível" colocaremos o valor do atributo border como 0 (zero).*

5.2.1 Criação dos elementos da tabela

Após a criação da tabela com sua devida formatação, precisamos inserir as células que receberão o conteúdo. Para isso utilizamos as *tags* **<th></th>**, para cabeçalho da tabela, ou seja, para que a informação contida na célula receba a formatação de negrito e centralizado, **<tr></tr>**, para a criação de linhas e **<td></td>** para a criação de colunas.

Dica: *Muitos dos atributos que se aplicam a tag <table> podem ser aplicados também a tag <td>, quando queremos que a formatação se aplique somente a célula desejada.*

Ao fim, a tabela criada ficará como no exemplo:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Tabela de exemplo</title>
      </head>
      <body>
            <table summary=”tabela” title=”Tabela de veículos” width="400" height=”200” align=”center” bgcolor=”#C0C0C0” bordercolor=”#FFFFFF” border="1" cellspacing="1" cellpadding="1" >
                  <tr>
                        <th>Carros</th><th>Valores</th>
                  </tr>
                  <tr>
                        <td>Gol</td><td>R$ 28.000,00</td>
                  </tr>
                  <tr>
                        <td>Corsa</td><td>R$ 15.000,00</td>
                  </tr>
                  <tr>
                        <td>Uno</td><td>R$ 25.000,00</td>
```

```
                </tr>
                <tr>
                      <td>Celta</td><td>R$ 17.000,00</td>
                </tr>
            </table>
      </body>
</html>
```

Note que o conteúdo da célula, seja um texto, uma imagem ou outro elemento fica entre as *tags* de abertura e fechamento da célula. Para alterar o alinhamento vertical do conteúdo das células, utiliza-se o atributo **valign** dentro das *tags* **<td></td>**, **<tr></tr>** ou **<th></th>**, que pode conter os valores **top**, **middle** ou **bottom**.

Obs.: *Mesmo que você não utilize mais tabelas para a estruturação do layout, certamente pode criá-las para organizar informações do site, como tabelas de preços.*

5.3 Layout por div

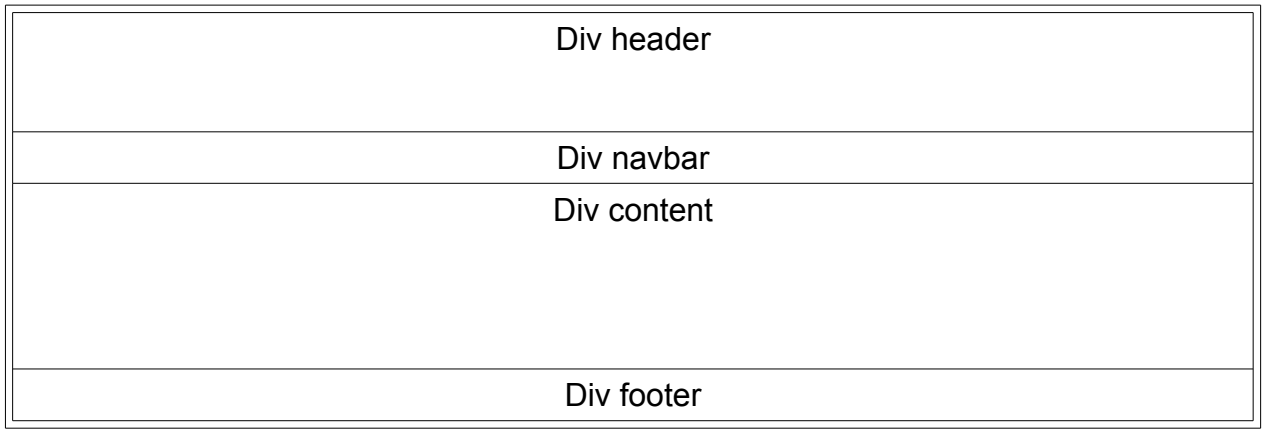
Div é uma *tag* que cria uma seção na página para inserirmos qualquer tipo de conteúdo, por isso é uma *tag* livre de semântica. Ela sozinha não poderá fazer muita coisa. Precisaremos de um nível mais avançado de **CSS** para criarmos um *layout* com a *tag* ** **. Por hora, veremos algumas propriedades.

Uma **div** é como uma célula de tabela flutuante, onde inserimos o conteúdo desejado entre a *tag* de abertura e fechamento. Um *layout* por **div** tem maior facilidade de atualização, maior dinamismo na estilização com folhas de estilo e uma variação de efeitos mais abrangente. Por exemplo, por **CSS3** hoje podemos arredondar os cantos de uma **div** ou colocar efeitos de sombreamento.

Nota: *Também utilizamos a tag para blocos de conteúdo, mas neste caso para conteúdos em linha.*

Para a estilização das **div's** por **CSS** precisaremos colocar o atributo **id** dentro de cada uma, para identificar e em seguida estilizar. Por padrão, geralmente criamos as seguintes **id's** para as **div's** do *layout*:

Div container



- A **div container** é aquela que vai conter todos os outros elementos, a que será centralizada e aplicada algum efeito de sombra, por exemplo;
- A **div navbar** é a barra de navegação do site, como no caso de um menu horizontal;
- A **div content** traz o conteúdo da página em questão. Normalmente conterá outras **div's** para separar o conteúdo;
- Criamos a **div footer** para o rodapé da página.

Obs.: *Você pode dar qualquer valor para o atributo id, esses valores aqui colocados são apenas senso comum.*

O código **XHTML** para esta página é simples, pois apenas inseriremos e identificar os objetos, serão as **CSS** que dirão seu tamanho, cor, alinhamento, posição e estilo. Veja o código **XHTML** da página:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Layout por div</title>
      </head>
      <body>
```

```
<div id="container">
        <!-- Aqui começa o topo -->
        <div id="header">
            <h1> Topo </h1>
        </div>
        <!-- Aqui termina o topo -->
        <!-- Aqui começa o menu -->
        <div id="navbar">
            <ul>
                <li><a href="index.html">Home</a></li>
                <li><a href="sobre.html">Sobre</a></li>
                <li><a href="produtos.html">Produtos</a></li>
                <li><a href="contato.html">Contato</a></li>
            </ul>
        </div>
        <!-- Aqui termina o menu -->
        <!-- Aqui começa o conteúdo -->
        <div id="content">
            <div id="esquerda">
                <h2>Notícia 1</h2>
                <p>O Consórcio World Wide Web (W3C) anunciou o lançamento de mais um escritório internacional, agora na Rússia. como resultado da crescente inserção e participação global no Consórcio, O Escritório está hospedado na Escola Superior de Economia (HSE) da Universidade Nacional de Pesquisa, fundada em 1992...</p>
                <p><a href="http://www.w3c.br/Noticias/W3cAbreEscritorioNaRussia">leia mais...</a></p>
            </div>
            <div id="centro">
                <h2>Notícia 2</h2>
                <p>Esta oitava edição do Prêmio Mario Covas conta com a parceria do W3C Brasil na categoria "Governo Aberto". O Prêmio Mário Covas é uma iniciativa da Secretaria de Estado de Gestão Pública de São Paulo e a participação do W3C Brasil inova por introduzir o reconhecimento de iniciativas de cidadãos que utilizam dados governamentais...</p>
```

```
                        <p><a
href="http://www.w3c.br/Noticias/OitavaEdicaoPremioMarioCovas">leia
mais...</a></p>
                    </div>




            <div id="direita">
                        <h2>Notícia 3</h2>
                        <p>O Consórcio World Wide Web (W3C) é um
consórcio internacional no qual organizações filiadas, uma equipe em tempo
integral e o público trabalham juntos para desenvolver padrões para a
Web...</p>
                        <p><a        href="http://www.w3c.br/Sobre">leia
mais...</a></p>
                    </div>
            </div>
            <!-- Aqui termina o conteúdo -->
            <!-- Aqui começa o rodapé -->
            <div id="footer">
                  <h3>&copy;    Copyright    -    Todos    os    direitos
reservados</h3>
            </div>
            <!-- Aqui termina o rodapé -->
        </div>
    </body>
</html>
```

Voltaremos a abordar este mesmo *layout* no final do capítulo **Entendendo as CSS**, portanto salve este arquivo como **estrutura_div.html** para que futuramente possamos voltar a desenvolver o *layout* por completo, uma vez tendo já adquirido nosso conhecimento com folhas de estilo.

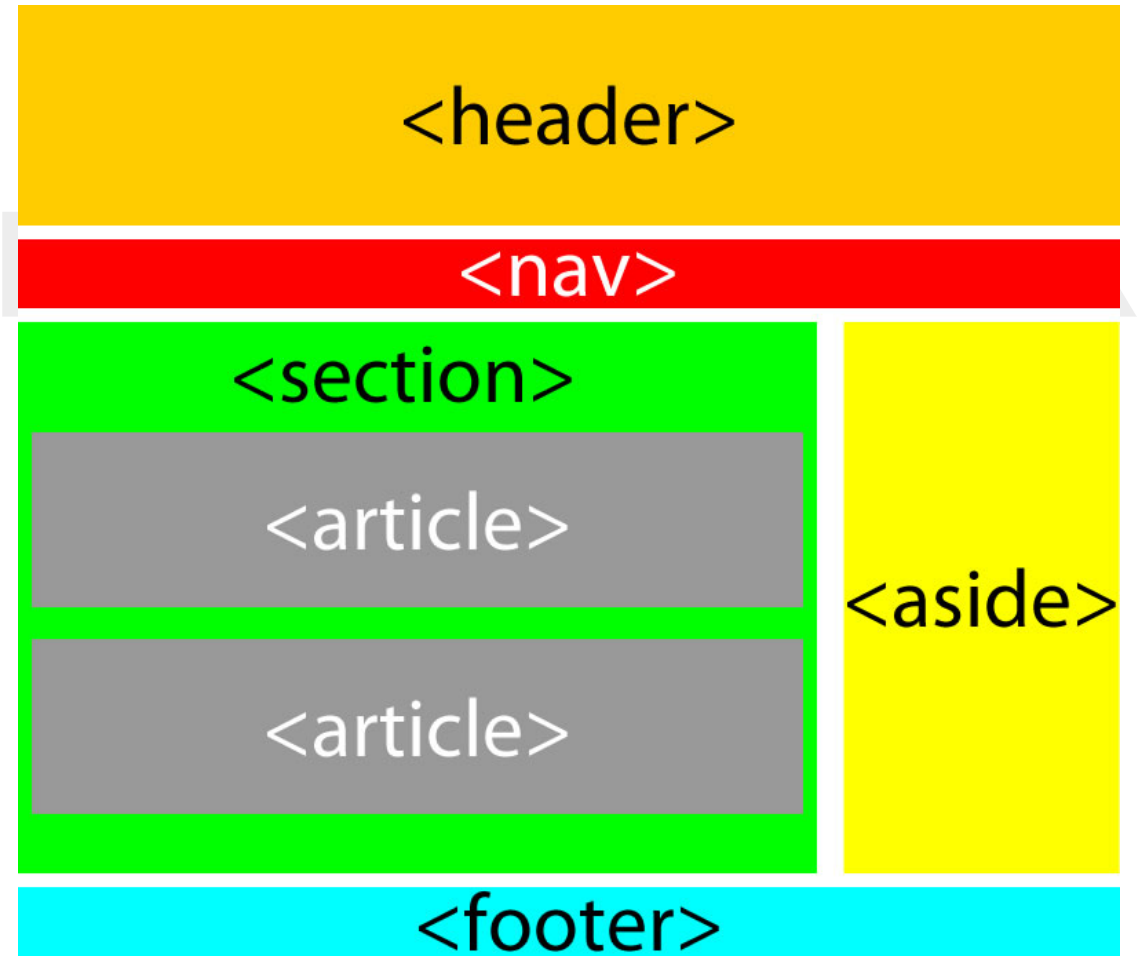
Todas as **div's** receberam **id** para futura estilização. As **div's** esquerda, centro e direita são para a digitação do conteúdo em três colunas. Na área do menu foram

inseridas *tags* para uma lista não ordenada onde faremos a estilização do mesmo. Os demais conteúdos receberam *tags* de cabeçalho e parágrafos.

	***Obs.:*** *Ainda no HTML5 é comum a utilização de div's para outros fins, mas para estruturação de layout teremos tags específicas para isso.*

5.4 Layout pelas novas tags HTML5

O **HTML5** criou *tags* para estruturação de *layout*, como a *tag* **<header></header>**, **<footer></footer>**, **<nav></nav>** e assim por diante. Desta forma, a própria *tag* inserida já possui formatações específicas e ideais para cada área do *layout*. Veja o exemplo:



Agora veja o código para formar uma página como essa, já com alguns conteúdos inseridos:

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

```
            <!-- HTML5 IE8-->
            <!--[if               lt               IE               9]><script
src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script><![endif]--
>
            <meta charset=”UTF-8” />
            <title>Layout por HTML5</title>
      </head>
      <body>
            <div id=”container”>
                  <!-- Aqui começa o topo -->
                  <header>
                        <h1>Topo</h1>
                        <h2>Slogan</h2>
                  </header>
                  <!-- Aqui termina o topo -->
                  <!-- Aqui começa o menu -->
                  <nav>
                        <ul>
                              <li><a  href="index.html"  alt="Página  Inicial"
title="Página Inicial">Home</a></li>
                              <li><a    href="sobre.html"    alt="Sobre    Nós"
title="Sobre Nós">Sobre</a></li>
                              <li><a     href="portfolio.html"     alt="Nossos
Projetos" title="Nossos Projetos">Portfólio</a></li>
                              <li><a href="localizacao.html" alt="Localização"
title="Localização">Localização</a></li>
                              <li><a  href="contato.html"  alt="Fale  Conosco"
title="Fale Conosco">Contato</a></li>
                        </ul>
                  </nav>
                  <!-- Aqui termina o menu -->
                  <!-- Aqui começa o conteúdo -->
                  <section>
                        <aside>
                              <h3>Links úteis</h3>
```

```
                        </aside>
                        <article>
                              <h3>Notícia 1</h3>
                              <h4>Lorem Ipsum é simplesmente uma simulação de
texto da indústria tipográfica e de impressos, e vem sendo utilizado desde o
século XVI, quando um impressor desconhecido pegou uma bandeja de tipos e os
embaralhou para fazer um livro de modelos de tipos. Lorem Ipsum sobreviveu
não só a cinco séculos, como também ao salto para a editoração eletrônica,
permanecendo essencialmente inalterado. Se popularizou na década de 60,
quando a Letraset lançou decalques contendo passagens de Lorem Ipsum, e mais
recentemente quando passou a ser integrado a softwares de editoração
eletrônica como Aldus PageMaker...</h4>
                              <p   class="leia"><a   href="noticia1.html">leia
mais...</a></p>
                        </article>
                        <article>
                              <h3>Notícia 1</h3>
                              <h4>Lorem Ipsum é simplesmente uma simulação de
texto da indústria tipográfica e de impressos, e vem sendo utilizado desde o
século XVI, quando um impressor desconhecido pegou uma bandeja de tipos e os
embaralhou para fazer um livro de modelos de tipos. Lorem Ipsum sobreviveu
não só a cinco séculos, como também ao salto para a editoração eletrônica,
permanecendo essencialmente inalterado. Se popularizou na década de 60,
quando a Letraset lançou decalques contendo passagens de Lorem Ipsum, e mais
recentemente quando passou a ser integrado a softwares de editoração
eletrônica como Aldus PageMaker...</h4>
                              <p   class="leia"><a   href="noticia1.html">leia
mais...</a></p>
                        </article>
                  </section>
                  <!-- Aqui termina o conteúdo -->
                  <!--Aqui começa o rodapé -->
                  <footer>
                        <h5>&copy; Copyright – Todos os direitos reservados –
Desenvolvido por: Elaborata Informática</h5>
                  </footer>
                  <!--Aqui termina o rodapé -->
            </div>
      </body>
```

```
</html>
```

Salve este projeto de *layout* como **layouthtml5.html** para que futuramente possamos estilizá-lo através das **CSS** e façamos as demais páginas do projeto.

Obs.: *Esta é apenas uma sugestão comum de uso, as tags de estruturação de layout do HTML5 podem ser utilizadas em qualquer local do documento.*

- A *tag* **<header></header>** define uma seção para o topo da página. Todo o conteúdo relativo ao topo deve ser inserido entre a *tag* de abertura e fechamento;
- A *tag* **<nav></nav>** é voltada para a barra de navegação ou menu;
- A *tag* **<section></section>** cria uma seção para todo o conteúdo;
- A *tag* **<article></article>** define uma área para inserção de notícias e blocos de conteúdo;
- A *tag* **<aside></aside>** é opcional para o caso de desejar criar um *sidebar*, como um segundo menu de categorias na página;
- A *tag* **<footer></footer>** cria uma seção para a inserção do rodapé da página.

O **W3C** em conjunto com os membros desenvolvedores do **HTML5** preocuparam-se em criar *tags* que facilitam a criação e manutenção da página, além de padronizar todas elas para uma mesma estrutura de *layout*. Elas serão estilizadas pelas **CSS** para estilização do seu tamanho, cor, estilo e posicionamento. Diferentemente das **div's** que são isentas de semântica, as novas *tags* **HTML5** têm a função de criar cada seção específica na estruturação do *layout*. Ainda podemos utilizar **div's** nesses *layouts*, como para inserção de elementos flutuantes, banners e propagandas, e ainda precisaremos da **div container** para conter todo o *layout* e centralizarmos futuramente o projeto.

Devemos sempre inserir duas linhas de código no início do cabeçalho da página para que as novas *tags* **HTML5** de estrutura funcionem no **Internet Explorer**:

```
<!-- HTML5 IE8-->
          <!--[if              lt              IE              9]><script
src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script><![endif]--
>
```

	***Dica:*** *As novas tags HTML5 são como as tradicionais, podem receber estilização por CSS, e insere-se seus conteúdos entre a abertura e fechamento.*

5.5 Iframes

Iframes são regiões ou seções criadas na página que exibem parte ou todo um conteúdo dentro de uma região específica. Por exemplo, poderíamos ter uma região inserida no nosso site onde é exibido parte do site da Elaborata, sendo como uma janela que nos trás diretamente a informação contida no site, não dependendo do local onde está inserido. Seguindo esta ideia, poderíamos inserir **iframes** no nosso site como a cotação do dólar ou previsão do tempo, por exemplo, sem a necessidade de atualizarmos esta informação. Veja:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Iframes</title>
      </head>
      <body>
            <iframe     name=”elaborata”     src=”http://www.elaborata.com.br”
width=”600”        height=”500”        scrolling=”auto”        frameborder=”0”
align=”center”></iframe>
      </body>
</html>
```

No exemplo, seria criado um **iframe** na página exibindo o site da Elaborata, com 600 pixels de largura e 500 pixels de altura, com barra de rolagem (**scrolling=”auto”**), sem borda e alinhado ao centro. Caso não queiramos barra de rolagem, basta definir o valor do atributo **scrolling** como “no” e caso queiramos borda, definiríamos o valor do atributo **frameborder** como “1”. Não é possível definir bordas maiores, apenas 0 para sem borda e 1 para com borda.

Digamos que se deseje criar um **link** em que ao clicarmos, a página de destino apareça dentro do **iframe**, tornando assim o **iframe** mais dinâmico e versátil. Para isso, definiremos o valor do atributo **target** do **link** com o nome do **iframe**. Veja o exemplo:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Iframes</title>
      </head>
      <body>
            <a     href=”http://www.elaborata.com.br”     target=”centro”>Site
Elaborata </a>
            <a href=”http://www.maujor.com” target=”centro”>Site Maujor </a>
            <a href=”http://pt.wikipedia.org” target=”centro”>Wikipedia </a>
            <iframe    src=””    name=”centro”    width=”500”    height=”500”
scrolling=”auto” frameborder=”0” align=”center”></iframe>
      </body>
</html>
```

5.5.1 Novos atributos HTML5 para iframes

O **HTML5** também trouxe implementações para o trabalho com **iframes**, vejamos:

5.5.2 Sandbox

Se utilizado sem valores, desabilita a inserção de conteúdos como *scripts*, formulários, *links* e *plugins*. Poderemos definir valores para habilitar cada recurso que gostaríamos como **allow-same-origin** para recursos de mesma origem, **allow-top-navigation** para *links*, **allow-forms** para permitir formulários e **allow-scripts** para *scripts*. Veja:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Iframes</title>
      </head>
      <body>
```

```
            <iframe src="http://www.elaborata.com.br" width="500" height="500"
sandbox=allow-top-navigation allow-forms allow-scripts></iframe>
      </body>
</html>
```

5.5.3 Seamless

Atributo *booleano* que faz com que o **iframe** tenha o mesmo contexto do documento, ou seja, regras de estilo e comportamento também são válidas para o **iframe**.

```
<!DOCTYPE html>
<html lang="pt-br">
      <head>
            <meta charset="UTF-8" />
            <title>Iframes</title>
      </head>
      <body>
            <iframe src="http://www.elaborata.com.br" width="500" height="500"
scrolling="auto" frameborder="0" align="center" seamless></iframe>
      </body>
</html>
```

5.5.4 Srcdoc

Insere uma marcação **HTML** dentro do **iframe**, muito utilizado para blogs por exemplo:

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

```
            <iframe seamless sandbox srcdoc="<p> Novos atributos HTML5 para
iframes:    <a    href="http://www.elaborata.com.br">    Faça    o    curso!
</a></p>"></iframe>
      </body>
</html>
```

Neste caso utilizamos **seamless** e **sandbox** como uma forma de proteção contra *scripts* e *malwares* que possam ser inseridos.

CAPÍTULO 6

LINKS

6.1 Links

Link significa ligação. Irá ligar um elemento inserido no site, como um texto, uma imagem ou um objeto à página que deve ser executada quando o usuário interagir com o objeto. Podemos inserir *links* em qualquer objeto do site, e para isso utilizaremos a *tag* **<a>**, ficando o elemento *linkado* entre as *tags* de abertura e fechamento. Essa *tag* sozinha não fará efeito algum, ela precisará de alguns atributos para ter sua função especificada. Veja:

Atributo da tag <a>	Função
href	Principal atributo que define o local para onde o usuário será enviado depois de interagir com o objeto *linkado*
target	Define em qual janela o *link* deve ser aberto. Os valores podem variar em: **_self**, para o *link* ser aberto na mesma janela, **_blank**, para o *link* ser aberto em outra janela ou **nome_qualquer**, para trabalhar com **iframes**
name	Utilizado em âncoras

Veja o exemplo:

```
<a href="http://www.elaborata.com.br" target="_blank" title="Site da Elaborata">Clique aqui</a>
```

Note que ao se definir *link*, o texto referente ao *link* recebe automaticamente uma formatação padrão de cor e estilo, que poderá ser alterada mais tarde pelas **CSS**.

6.1.1 Tipos de links

Dos tipos de *link* existentes, veremos os quatro mais usualmente usados nos dias de hoje:

- *Link* Relativo

Um *link* que tem como página de destino uma página de navegação interna do site. Para criá-lo, damos como valor do atributo **href** o caminho e o nome do arquivo desejado. Exemplo:

```
<a href="contato.html" target="_self" title="Fale Conosco">Contato</a>
```

Em geral, utilizamos o valor do atributo **target** como **_self** em *links* relativos, para que o usuário permaneça na mesma janela do navegador ao interagir com o *link*.

- *Link* Absoluto

Link para uma página externa ao site. Neste caso, como valor do atributo **href** damos o endereço completo da página de destino. Exemplo:

```
<a href="http://www.elaborata.com.br" target="_blank" title="Site da Elaborata">Cursos na área de TI</a>
```

Em geral para esses casos damos ao atributo **target** o valor **_blank**, para que a página de destino seja aberta em outra janela e o usuário continue com a navegação no site atual.

- *Link* de *download*

Link para fazer o *download* de um arquivo. Semelhante ao *link* relativo, mas como valor do atributo **href** temos um arquivo que não contém uma extensão **.htm** ou **.html**, mas alguma outra. Exemplo:

```
<a href="musica.mp3" title="Faça o Download de Música">Download</a>
```

Neste caso não precisamos definir **target**, já que a página não será aberta, mas sim será destinado o *download* do arquivo ao usuário.

- *Link* de âncora

Uma **âncora** é um *link* para a mesma página, mas em outro ponto da barra de rolagem, para casos como artigos extensos onde ao final poderíamos encontrar um *link* que já nos enviaria de volta para o topo da página. Para isso, precisaremos do atributo **name**. Ele irá marcar o ponto de retorno na barra de rolagem quando o usuário clicar no *link*. Veja o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
	<head>
		<meta charset="UTF-8" />
		<title>Link de Âncora</title>
	</head>
	<body>
		<a name="topo"></a>
		<br /><br /><br /><br /><br /><br /><br /><br /><br />
<br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
<br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
```

```
<br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
<br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
<br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
            <a href="#topo"> TOPO </a>
      </body>
</html>
```

O valor do atributo **name** pode ser qualquer nome, sem espaço ou caracteres especiais. Este nome é referenciado no *link* ao final do documento, precedido por uma cerquilha (**#**), e assim o navegador compreende que é um *link* para a mesma página, fazendo a ligação entre o valor do atributo **name** no início com o valor do atributo **href** ao fim do documento.


ELABORATA
INFORMÁTICA
CAPÍTULO 7
FORMULÁRIOS

7.1 Trabalhando com formulários

Sempre que quisermos que o usuário possa interagir com a página *web*, preenchendo um formulário de contato, criando um cadastro ou fazendo um login precisaremos das *tags* de criação de formulários para este fim. Para isso, precisaremos do **XHTML** que dará a base para as inserções de dados ao usuário, um agente de processamento do formulário, que é geralmente um arquivo em **PHP**, que irá executá-lo e um **CSS** para deixá-lo esteticamente mais agradável.

A *tag* para criação de formulários é a *tag* **<form></form>**. Todos os elementos do formulário deverão ficar entre as *tags* de abertura e fechamento.

```
<form name="exemplo_contato" action="form.php" enctype="application/x-www-form-urlencoded" method="post" >
    <p>
        <label for="nome">Nome: </label>
        <input name="nome" type="text" />
    </p>
</form>
```

Como podemos visualizar, os demais elementos filhos estarão sempre entre o início e fim de **<form></form>**. A *tag* para início de formulário deve ter alguns atributos para um bom funcionamento. São eles:

Atributo da tag <form></form>	Função
name	Atribui nome ao formulário
action	O valor desse atributo será o arquivo executado quando o usuário clicar no botão de envio (*submit*). Em geral, usasse um **PHP**
enctype	Define o tipo de dados. Os valores variam entre **application/x-www-form-urlencoded** como padrão, e **multipart/form-data** para formulários com campos para *upload* de arquivos
method	Método de envio das informações, podendo ter como valores **GET** ou **POST**, sendo mais comumente utilizado o atributo **POST**, pois ele encapsula os dados e envia

A *tag* que irá inserir grande parte dos objetos de formulário é a **<input />**, mudando apenas o tipo de campo através do atributo **type**. Trata-se de uma *tag* de elemento vazio. Conheça algumas variações de campos **input**:

- *Text* (Campo de texto)

```
<input name="nome" type="text" title="Nome" value="valor inicial" size="50" maxlength="20" />
```

Um dos campos mais utilizados. Cria um campo de texto padrão para valores genéricos. O atributo **name** define o nome do campo para que o **PHP** possa reconhecê-lo. O atributo **type** define o tipo de **input**. O atributo global **title** está definindo o texto que aparecerá quando posicionarmos o mouse sobre o campo. O atributo **value** define um valor inicial de preenchimento assim que a página é carregada. O atributo **size** define a largura do campo. O atributo **maxlength** define o número máximo de caracteres.

- *Password* (Senha)

```
<input name="senha" type="password" title="Digite sua senha" size="50" maxlength="15" />
```

Serve para criar um campo de senha, onde os caracteres digitados são ocultos. Os atributos são semelhantes ao valor **text**.

- *Checkbox* (Caixa de seleção)

```
<input name="cursos" type="checkbox" value="html5" title="Curso de HTML5" checked="checked" />
```

Para criarmos caixas onde é possível escolher mais de uma opção. O atributo **value** indica o valor que será processado. O atributo **checked** serve para que se tenha um estado inicial como marcado. Em **HTML** ele é um atributo *booleano* e não há a necessidade de se inserir um valor, em **XHTML** insere-se um valor idêntico ao atributo. Caso não queiramos que a caixa tenha um valor inicial marcado, basta não inserir o atributo.

- *Radio* (Botão de opção)

```
<input name="sexo" type="radio" title="Masculino" value="masculino" checked="checked" />
```

Botão onde apenas uma opção pode ser escolhida. Para que o efeito funcione, todos os botões da mesma categoria devem ter o mesmo valor do atributo **name**, variando apenas em **value**.

- *File* (Arquivo)

```
<input name="anexar" type="file" size="50" title="Procurar arquivo..." />
```

Valor para que seja possível anexar arquivos no envio do formulário. Ao se utilizar este valor no formulário, devemos mudar o **enctype** para **multipart/form-data**.

- *Button* (Botão)

```
<input type="button" value="Botão" title="Clique aqui" />
```

Cria um botão genérico. Geralmente utilizado para ganhar funcionalidades por eventos **Java Script**.

- *Submit* (Enviar)

```
<input type="submit" value="Enviar" title="Enviar" />
```

Cria um botão que executará o que foi definido no atributo **action** da *tag* **<form></form>**.

- *Reset* (Limpar)

```
<input type="reset" value="Limpar campos" title="Apagar dados" />
```

Cria um botão para apagar todos os dados digitados no formulário.

- *Image* (Imagem)

```
<input type="image" src="imagem.jpg" title="Enviar dados" />
```

Utilizado para se inserir uma imagem de um botão ao invés do botão padrão **XHTML**.

Outras *tags* comumente utilizadas em formulários:

- *Tag* **<select></select>**

Para definirmos uma lista de opções. As opções serão definidas pela *tag* **<option></option>**:

```
<select name="cursos">
      <option>Escolha um curso</option>
      <option value="XHTML">XHTML</option>
      <option value="CSS">CSS</option>
      <option value="PHP">PHP</option>
</select>
```

- *Tag* **<textarea></textarea>**

Cria um campo de texto com mais linhas de preenchimento, para áreas como comentários por exemplo.

```
<textarea name=”mensagem” cols=”50” rows=”4”></textarea>
```

Cols e **rows** definem a quantidade de linhas e colunas que a área de texto terá respectivamente.

- *Tag* **<fieldset></fieldset>**

Separa as informações do formulário em categorias de campos. A *tag* **<legend></legend>** define um título para as categorias.

```
<fieldset>
      <legend>Sexo</legend>
            <input name=”sexo” type=”radio” value=”masculino” /> Masculino
            <input name=“sexo” type=”radio” value=”feminino” /> Feminino
</fieldset>
```

Obs.: *Não se esqueça de sempre definir name ou value para os atributos de formulário, para que o PHP possa processá-lo corretamente.*

Veja o exemplo de um formulário semântico com algumas das *tags* aprendidas:

Nota: *Não é obrigatoriamente necessário a definição da tag <label></label> para cada campo de formulário, mas é semântico que se faça assim, para uma melhor separação dos elementos e compreensão do formulário.*

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Formulário de contato</title>
      </head>
      <body>
            <form                                                name=”formulario”
action=”http://www.elaborata.com.br/teste/aula.php”                method=”post”
enctype=”application/x-www-form-urlencoded”>
                  <fieldset>
                        <legend>Dados pessoais</legend>
                        <p>
                              <label for=”nome”>Nome: </label>
```

```
                              <input   type="text"   name="nome"   title="Nome"
size="60" />
                        </p>
                        <p>
                              <label for="e-mail">E-mail: </label>
                              <input  type="text"  name="e-mail"  title="E-mail"
size="60" />
                        </p>
                        <p>
                              Sexo:
                              <label><input      name="sexo"      type="radio"
value="masculino" /> Masculino </label>
                              <label><input      name="sexo"      type="radio"
value="feminino" /> Feminino </label>
                        </p>
                  </fieldset>
                  <fieldset>
                        <legend>Cursos:</legend>
                        <p>
                              <label><input    name="cursos"    type="checkbox"
value="webstandards" /> Webstandards </label>
                              <label><input    name="cursos"    type="checkbox"
value="dreamweaver" /> Dreamweaver </label>
                              <label><input    name="cursos"    type="checkbox"
value="fireworks" /> Fireworks </label>
                              <label><input    name="cursos"    type="checkbox"
value="flash" /> Flash </label>
                        </p>
                  </fieldset>
                  <p>
                        <label for="mensagem">Mensagem: </label>
                  </p>
                  <p>
                        <textarea    name="mensagem"    rows="6"    cols="50"
title="Digite sua mensagem"></textarea>
                  </p>
```

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

7.2 Novas funções para formulários com HTML5

Foram criadas novas funções para facilitar o uso dos formulários no **HTML5**, e ajudar em sua validação sem a necessidade de *scripts* para isso. Vejamos:

Obs.: *O Internet Explorer não suporta nenhuma nova implementação para formulários em HTML5.*

7.2.1 Atributo form

Associa um elemento de formulário inserido fora das *tags* **<form></form>** ao formulário em questão. O valor do atributo **form** deve ser igual a uma id aplicada a *tag* **<form></form>**:

```
<form method="post" action="form.php" id="formcontato">
      Nome: <input name="nome" type="text" title="Nome" /> <br />
      E-mail: <input name="email" type="text" title="E-mail" /> <br />
      <input type="submit" value="Enviar" /> <br />
</form>

Mensagem: <textarea name="mensagem" form="formcontato"></textarea>
```

No exemplo, o que for preenchido no campo de mensagem será também processado quando o usuário utilizar o botão de envio.

7.2.2 Atributos para campo numérico

Para criarmos um campo numérico poderemos utilizar o atributo **number** para a *tag* **<input />**.

```
Quantidade: <input name="quantidade" type="number" />
```

Podemos também definir um valor inicial, máximo e mínimo para o campo:

```
Quantidade: <input name="quantidade" type="number" min="1" max="10" value="5" />
```

Os atributos numéricos atualmente também não funcionam no **Mozilla Firefox**.

7.2.3 Atributo range

O atributo **range** para a *tag* **<input />** cria um controle deslizante para uma escolha numérica. Não é suportado pelo **Mozilla Firefox**:

```
Quantidade: <input name="quantidade" type="range" min="10" max="50" value="20" step="1" />
```

O atributo **step** demonstra que ao deslizar o controle o efeito deverá ocorrer de 1 em 1.

7.2.4 Atributo autocomplete

Os navegadores por padrão guardam os dados digitados em formulários anteriormente, e quando o usuário dá foco ao campo ele lhe traz sugestões baseado em seu histórico. Com **HTML5** poderemos ativar ou desativar este recurso com o atributo **autocomplete**. Ele possuirá dois valores, *on* e *off*, sendo *on* o padrão e não havendo necessidade de definir. Este atributo se aplica as *tags* **<form></form>** e **<input />**. O **Apple Safari** não tem suporte para este atributo.

```
<input name="nome" type="text" autocomplete="off" />
```

7.2.5 Atributo placeholder

Oferece uma dica sobre o que deve ser preenchido dentro do campo. Pode ser inserido nas *tags* **<input />** e **<textarea></textarea>**:

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

7.2.6 Atributo autofocus

Define o objeto que já entrará em foco assim que a página for carregada. Trata-se de um atributo *booleano.* Aplica-se as *tags* **<button></button>**, **<input />**, **<select></select>** e **<textarea></textarea>**:

```
Nome: <input name="nome" type="text" title="Nome" placeholder="Digite seu nome completo" autofocus />
```

7.3 Validação de formulário

Validar um formulário é torná-lo mais seguro sobre o conteúdo que será inserido pelo usuário, já que em um formulário o usuário poderia enviar *scripts* maliciosos para o servidor. Também facilita o preenchimento de cada campo. Poderemos através da validação definir campos apenas numéricos, campos de e-mail, de uma quantidade específica de caracteres e campos obrigatórios.

Esta validação até os dias de hoje sempre foi feita através de **JavaScript**, o que não o tornava tão seguro caso o usuário desabilitasse o uso de *scripts* nas configurações do navegador, mas agora também pode ser feita por **HTML5**, tornando fácil e dinâmico a criação de uma validação. Vejamos:

7.3.1 Atributo required

Se utilizado torna o campo de preenchimento obrigatório. Trata-se de um atributo *booleano*, ou seja, não precisa de valor. Ele verifica apenas se o campo foi ou não preenchido, e caso não seja o navegador apresentará ao usuário uma mensagem padrão informando a necessidade do preenchimento. Pode ser inserido nas *tags* **<input />**, **<select></select>** e **<textarea></textarea>**:

```
Nome: <input name=”nome” type=”text” required />
```

7.3.2 Atributo email

Verifica se no campo inserido foi digitado um valor que corresponda a um e-mail, ou seja, verifica se foi digitado ao menos a arroba. Trata-se de um novo tipo para a *tag* **<input />**, e caso o usuário não digite um valor semelhante a um endereço de e-mail o navegador apresentará uma mensagem padrão informando a necessidade do preenchimento.

```
Nome: <input name=”e-mail” type=”email” />
```

Um valor como a@b.c passará na validação.

Veja o exemplo de um formulário completo utilizando os novos atributos **HTML5**. Salve como **contato.html** para que futuramente possamos estilizá-lo e inseri-lo em uma página:

```
<!DOCTYPE html>
<html lang=”pt-br”>
      <head>
            <meta charset=”UTF-8” />
            <title>Formulário de contato</title>
      </head>
      <body>
            <form     name=”formulario”     action=”form.php”     method=”post”
enctype=”application/x-www-form-urlencoded”>
                  <fieldset>
                        <legend>Dados pessoais</legend>
                        <p>
                              <label for=”nome”>Nome: </label>
                              <input   type=”text”   name=”nome”   title=”Nome”
size=”60” placeholder=”Digite seu nome completo” required autofocus />
                        </p>
                        <p>
                              <label for=”e-mail”>E-mail: </label>
                              <input type=”email” name=”e-mail” title=”E-mail”
size=”60” placeholder=”exemplo: contato@empresa.com.br” required />
                        </p>
                        <p>
                              <label for="estado">Estado: </label>
                              <select name="estado">
                                    <option    selected="selected">Escolha    um
Estado</option>
                                    <option value="Paraná">PR</option>
                                    <option value="Santa Catarina">SC</option>
                                    <option      value="Rio      Grande      do
Sul">RS</option>
                              </select>
                        </p>
                  </fieldset>
                  <fieldset>
                        <legend>Produtos: </legend>
```

```
                        <p>
                              <label><input   name="produtos"   type="checkbox"
value="Camiseta" /> Camiseta </label>
                              <label>Quantidade</label>
                              <input  name="quantidade"  type="number"  min="1"
max="10" value="1" />
                        </p>
                        <p>
                              <label><input   name="produtos"   type="checkbox"
value="Boné" /> Boné </label>
                              <label>Quantidade</label>
                              <input  name="quantidade"  type="number"  min="1"
max="10" value="1" />
                        </p>
                        <p>
                              <label><input   name="produtos"   type="checkbox"
value="Adesivo" /> Adesivo </label>
                              <label>Quantidade</label>
                              <input  name="quantidade"  type="number"  min="1"
max="10" value="1" />
                        </p>
                  </fieldset>
                  <p>
                        <label for=”mensagem”>Mensagem: </label>
                  </p>
                  <p>
                        <textarea     name=”mensagem”     rows=”6”     cols=”50”
title=”Digite   sua   mensagem”   placeholder=”Digite   aqui   sua   mensagem”
required></textarea>
                  </p>
                  <p>
                        <input type=”submit” value=“Enviar” title=”Enviar” />
                  </p>
            </form>
      </body>
</html>
```


ELABORATA
INFORMÁTICA
CAPÍTULO 8
ENTENDENDO AS CSS

WEBSTANDARDS COM HTML5

8.1 Entendendo as CSS

CSS é a sigla para **Cascading Style Sheets** ou **Folhas de Estilo em Cascata** e servem para a estilização de documentos **XHTML**. Ela não terá a função de inserir elementos, e sim personalizar e formatar os elementos já inseridos no código **XHTML**. Através das folhas de estilo poderemos alterar toda a aparência do site, e manter um padrão para que toda a página siga a mesma formatação. Desta forma economizaremos tempo e desgaste no momento da criação e atualização das páginas.

Existem três formas de inserirmos as folhas de estilo no documento:

- *In-line*;
- Incorporado;
- Linkado externamente.

O **CSS In-line** é escrito dentro da *tag*, dando a ela uma estilização válida somente para aquele elemento. Para isso, utilizamos o atributo global **style**. Veja o exemplo:

```
<p style="color:#900; font-weight:bold;">Parágrafo estilizado por CSS in-line</p>
```

No exemplo, o texto receberia a formatação de negrito e com a cor vermelha.

Já o **CSS Incorporado** é escrito todo na seção **<head></head>** do documento, e ele especifica a estilização para toda aquela página. Essa formatação é feita através da *tag* **<style></style>**. Veja o exemplo:

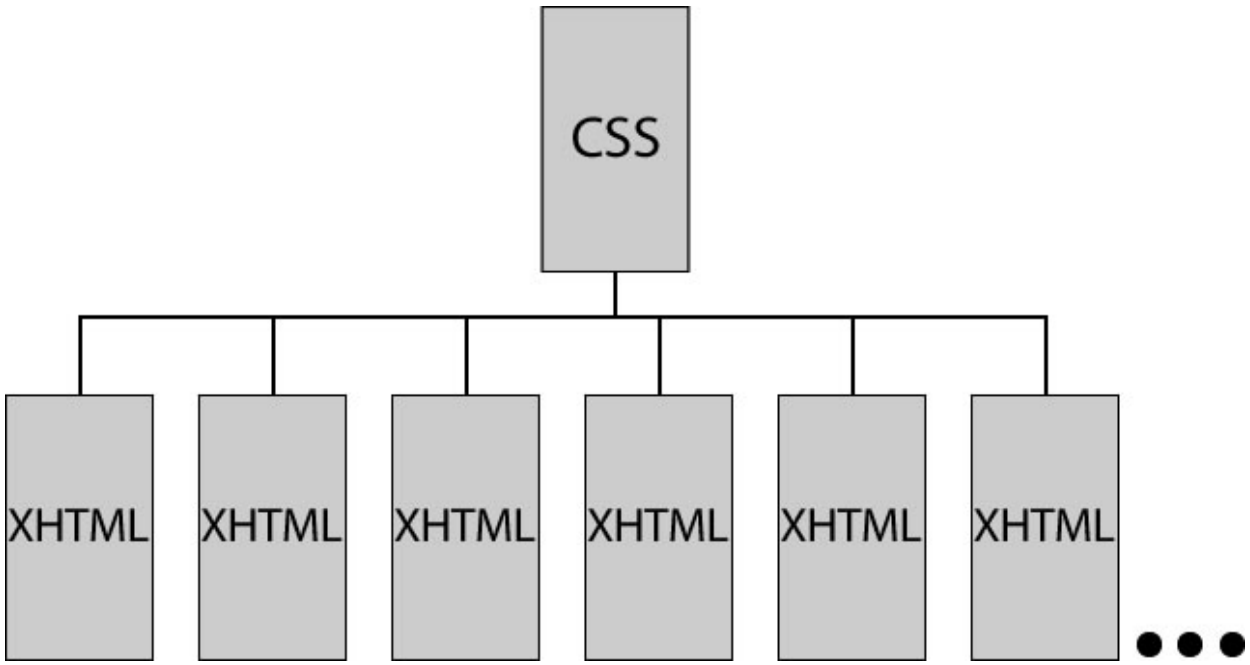
```
<!DOCTYPE html>
<html lang="pt-br">
      <head>
            <meta charset="UTF-8" />
            <title>Formulário de contato</title>




      <style type="text/css">
                  p {
```

```
                        color: #900;
                        font-weight: bold;
                }
            </style>
        </head>
        <body>
            <p>Parágrafo estilizado por CSS incorporado</p>
        </body>
</html>
```

Podemos observar que foi declarado uma *tag* **<style></style>** na seção **head** do documento, com um atributo **type**. Este atributo é facultativo de uso em **HTML5**. Em seguida, temos a identificação do elemento a ser atingido através da **CSS**, no caso a *tag* ** **. Na sequência, entre chaves, temos a declaração das propriedades e valores a serem configurados para esta *tag*.

Dica: *No HTML5 podemos inserir a tag <style></style> para estilização em qualquer região do documento, não apenas na seção <head></head>.*

No caso do **CSS externo** ou **linkado** criaremos uma única folha de estilo que, ao ser ligada a diversas páginas, terá o poder de estilizar todas elas ao mesmo tempo, pois todas dependerão dela para sua formatação. A imagem a seguir ilustra a ideia:



Este arquivo **CSS** linkado terá a extensão **.css**. Faremos esta ligação das páginas **XHTML** com o arquivo **CSS** de estilização através da *tag* **<link />** vista anteriormente, na seção **<head></head>** do documento, e ela deve ser inserida em cada um dos documentos **XHTML** em que se deseja a estilização por meio desta folha de estilos. Veja o exemplo:

```
<!DOCTYPE html>
<html lang="pt-br">
      <head>
            <meta charset="UTF-8" />
            <title>Formulário de contato</title>
            <link rel="stylesheet" type="text/css" href="estilos/style.css" />
      </head>
      <body>
            <p>Parágrafo estilizado por CSS incorporado</p>
      </body>
</html>
```

Todos esses atributos para a *tag* **<link />** são obrigatórios em **HTML 4.01**. Os atributos **rel** e **href** são obrigatórios em **HTML5** e o atributo **type** é facultativo. O atributo **rel** mostra a relação da *tag* **<link />**, definido como folhas de estilo, o **type** o tipo de documento linkado e o atributo **href** indica o caminho e o nome do arquivo que contém as estilizações (no exemplo, o nome do arquivo **style.css** contido dentro da pasta **estilos**). Tome muito cuidado ao fazer o *upload* da página para o servidor em não alterar este caminho, para que não aconteça do código-fonte não mais encontrá-lo. É preferível que se utilizem programas de **FTP** para envio da página com maior segurança.

As folhas de estilo são chamadas de cascata devido a uma hierarquia existente entre elas. Digamos que temos um documento conflitante, onde em uma **CSS in-line** pedimos para determinada *tag* tenha seu texto estilizado em vermelho, uma **CSS incorporada** pedindo para que a mesma *tag* tenha seu texto estilizado em azul, e uma **CSS linkada** pedindo para que a mesma tenha o seu texto estilizado em verde. Qual prevalecerá? Para resolver esta questão foi criada uma hierarquia, que será sempre de dentro para fora, ou seja, a estilização que está mais perto da *tag* (**in-line**) prevalecerá

sobre as outras e assim suscetivamente. Caso não haja **CSS in-line**, prevalecerá a **incorporada**, caso não haja nenhuma outra, predominará a **CSS linkada**. Essa hierarquia é útil para momentos em que gostaríamos que todo o site tivesse uma determinada estilização, mas com alguma exceção, neste caso inseriríamos uma **CSS** mais local para o elemento específico.

Obs.: *Se algum valor CSS for declarado com a definição !important ele terá prioridade sobre os demais, ignorando o efeito cascata.*

8.2 Sintaxe CSS

Observe que as **CSS** são escritas de forma diferente que o **XHTML**, com a presença de **dois-pontos**, **chaves** e **ponto-e-vírgula**. Chamamos o elemento que queremos atingir no código **XHTML** de **seletor**, o conteúdo entre as chaves de **declaração**, as definições do que será feito de **propriedades** e as especificações das propriedades de **valores**, desta forma:

```
seletor {
      propriedade: valor;
}
```

Os espaçamentos são apenas por questão de indentação. Perceba que ao final de cada propriedade da declaração temos um dois-pontos, e ao fim de cada valor, um ponto-e-vírgula para mostrar que concluímos a formatação daquela propriedade. Tudo que estiver entre chaves (declaração) atingirá a *tag* ou elemento definido no seletor. Poderemos definir quantas propriedades julgarmos necessárias para cada seletor, desde que sigamos corretamente a sintaxe **CSS**. É possível utilizar qualquer propriedade para atingir qualquer elemento, mas em alguns casos não fará sentido.

Veja um exemplo de um arquivo **CSS linkado** estilizando uma página **XHTML**:

```
body {
      margin: 0;
      background-color: #CCC;
      color: #FFF;
      }
h1 {
      font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
```

```
	color: #900;
	font-size: 26px;
}
h2 {
	font-weight: normal;
	font-size: 22px;
	color: #393;
}
p {
	font-size: 16px;
}
```

No exemplo, foram atingidas as *tags* **body**, **h1**, **h2** e **p** do código **XHTML**. Receberam formatações como cor de fundo da página, margens e formatações de texto. Como é um arquivo linkado, se linkassemos vários arquivos **XHTML** a esta mesma folha de estilos, todos eles receberiam a mesma formatação padrão definida na **CSS**. Atualizar a aparência do site todo também ficaria muito mais fácil, podendo mudar a cor de fundo de todas as páginas, por exemplo, alterando apenas uma única linha de código.

Note que no valor 0 (zero) para as margens não foi definido uma unidade de medida, já nas demais foi definido px como padrão para pixels. Nas **CSS** sempre precisaremos definir a unidade de medida ao qual os números se referem, que podem ser **px** (pixels), **%** (porcentagem), **cm** (centímetro), **pt** (ponto) ou **in** (polegada), a menos que o valor seja 0 (zero).

8.3 Agrupamentos

Podem haver casos em que gostaríamos que várias *tags* **XHTML** recebessem a mesma formatação **CSS**, por exemplo, que todos os cabeçalhos da página tenham o mesmo tipo de letra, variando somente em cores e tamanhos. Para isto, definimos todas as *tags* que gostaríamos de atingir no mesmo seletor, separando por vírgula:

```
h1, h2, h3, h4, h5, h6 {
	font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
	}
```

Note que a fonte *Trebuchet MS* foi colocada entre aspas. Isso deve ocorrer sempre que um valor para a propriedade for um conteúdo que tenha espaços em branco em sua descrição. Outra coisa interessante é que temos a declaração de várias fontes, uma família. Isso ocorre devido a não termos certeza que o usuário que acessará a página terá a primeira fonte escolhida instalada no seu sistema. Então por precaução temos algumas fontes que assumirão o lugar de sua predecessora caso ela não exista no computador do usuário, substituindo-a, evitando assim frustração de quem acessa a página. Todo o caso, vale lembrar que devemos sempre utilizar fontes-padrão, que são fontes que se julga que devam existir em todos os sistemas atuais.

8.4 Herança

Outro momento, que chamamos de **herança**, é quando temos a definição de elementos sequenciais, ou seja, uma *tag* está inserida dentro de outra. Se quiséssemos atingir os itens de uma lista ordenada dentro de um parágrafo, neste caso separaríamos os elementos declarados no seletor por um espaço em branco:

```
p ul li {
      display: inline;
      }
```

No exemplo, pedimos que todo o item que surgir em uma lista e estiver dentro de algum parágrafo, seja exibido em linha, e não em bloco. Chamamos de **herança** pois o texto receberá todas as formatações já aplicadas a *tag* ** **, as formatações de **
** e de **
, ou seja, vai herdando da *tag* "pai" os atributos aplicados anteriormente. A **herança não ocorre quando temos uma formatação específica para o elemento filho, por exemplo:

```
p {
      color: #900;
      }
p ul li {
      color: #CCC;
}
```

Neste caso, somente receberão a primeira cor aqueles elementos dentro da *tag* ** ** não contidos em listas, pois os contidos em listas receberão a segunda cor. Se não tivéssemos definido uma formatação específica para listas, todos os textos contidos

em *tag* **
** e **
** também receberiam a mesma formatação por **herança**, julgando que estejam contidas em um parágrafo.

8.5 Propriedades para espaçamentos

Vamos ver algumas questões importantes a serem tratadas no estudo das **CSS**. Primeiro tópico importante: existe um elemento-chave, que atingirá todas as *tags* do documento, representado por um asterisco. Geralmente usa-se este elemento para remover já de antemão bordas, margens e espaçamentos, veja seu uso:

```
* {
      margin: 0;
      padding: 0;
      border:0;
      vertical-align: baseline;
}
```

No exemplo foram zerados valores como margens e espaçamentos. Isso é chamado pelos profissionais de *reset*.

Margem é o espaçamento dado à área externa do objeto aonde é inserida a propriedade. Atingiremos as margens do seletor com a definição da propriedade **margin**. No caso de margens temos várias formas de trabalhar. Primeiro, especificando valores para cada uma delas:

```
body {
      margin-top: 5px;
      margin-right: 3px;
      margin-bottom: 10px;
      margin-left: 3px;
}
```

Note que especificamos valores para cada uma das margens individualmente, sendo *top* a margem superior, *right* a margem direita, *bottom* a margem inferior e *left* a margem esquerda. Também poderíamos atingi-las em uma única propriedade:

```
body {
      margin: 5px 3px 10px 3px;
}
```

A ordem para inserção é sempre a mesma e lembra os ponteiros de um relógio, sendo o primeiro valor sempre se referindo ao *top*, o segundo se referindo ao *right*, o terceiro se referindo ao *bottom* e por último o *left*.

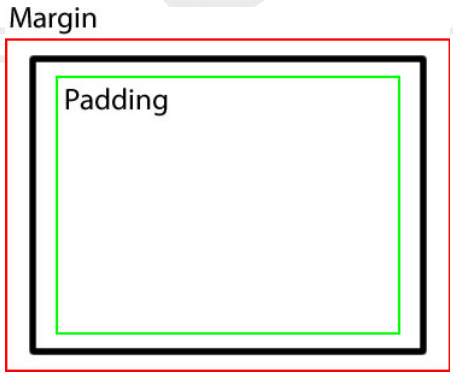
	***Dica:*** *Se definirmos margin: 0 auto para um elemento, ele centralizará horizontalmente onde estiver inserido, pois define zero para o top, e o default centralizado para as demais.*

Quando os valores são os mesmos, a especificação de apenas um valor aplicará esse valor às quatro margens:

```
body {
      margin: 3px;
}
```

No exemplo, todas as margens terão o valor de 3 pixels.

Outro espaçamento comumente utilizado é o **padding**, que define o espaçamento entre o limite do elemento e o conteúdo nele inserido, como no caso de tabelas e div's. Trabalhar com **padding** é semelhante às margens em suas propriedades e valores. O gráfico a seguir ajuda a entender a diferença entre **margin** e **padding**:



Utilizaremos **margin** quando queremos alterar o espaçamento entre os limites do conteúdo e sua área externa, e **padding** quando gostaríamos de alterar o espaçamento entre os limites do conteúdo e sua área interna.

8.6 Propriedades para fundo

As propriedades **CSS** para alterar o fundo são usualmente utilizadas para alterar o fundo da página, neste caso aplicado a *tag* **body**, mas também podem ser utilizadas para atingir outros elementos como tabelas, div's ou seções.

Veja uma tabela prática com as principais propriedades para alteração de fundo:

Propriedade	Função	Valores
Background-color	Cor de fundo	Código hexadecimal da cor ou nome da cor em inglês
Background-image	Imagem de fundo	url(caminho/imagem.jpg)
Background-repeat	Tipo de repetição da imagem de fundo	No-repeat, para que não repita, repeat-x, para que repita na horizontal ou repeat-y, para que repita na vertical
Background-position	Posição da imagem de fundo	Um primeiro valor numérico para a distância em relação à esquerda do elemento inserido e um segundo valor numérico para a distância em relação ao topo do elemento inserido
Background-attachment	Rolagem do fundo	Fixed para que a imagem não role com o documento

Exemplo:

```
body {
	background-color: #CCC;
	background-image: url(imagens/fundo.png);
	background-repeat: no-repeat;
	background-position: 400px 100px;
	background-attachment: fixed;
}
```

Também poderemos escrever todos os valores de forma resumida, desde que se siga a ordem correta de declaração (**color image repeat attachment position**):

```
body {
	background: #CCC url(imagens/fundo.png) no-repeat 400px 100px fixed;
}
```

Neste exemplo teremos as mesmas propriedades do anterior, mas com menor necessidade de código escrito para isso. Caso não vá utilizar todos os valores para a propriedade, basta pular para o próximo valor.

	***Dica:*** *O código hexadecimal de cores pode ser resumido no CSS em casos de cores em que o código se repete na definição de cada par. Por exemplo, poderíamos escrever #CC0099 como #C09.*

8.7 Propriedades para bordas

Podemos alterar pelas folhas de estilo as bordas de uma tabela, imagem, div ou outros elementos. Para isso, utilizamos essas principais propriedades **CSS**:

Propriedade	Função	Valores
Border-width	Largura da borda	Valor numérico para largura. É comum definir 0 (zero) para links em imagens por questões estéticas
Border-color	Cor da borda	Código hexadecimal da cor ou nome da cor em inglês
Border-style	Estilo da borda	Dashed, dotted, ridge, double, solid são alguns exemplos.

Exemplo:

```
div {
	border-width: 3px;
	border-color: #900;
	border-style: dashed;
}
```

Ou pode ser escrita em uma única linha, seguindo a ordem (**size style color**):

```
div {
	border: 3px dashed #900;
}
```

Se quisermos também poderemos alterar cada borda de forma única, por exemplo, apenas a borda de baixo, ou apenas a da esquerda e direita, definindo qual borda gostaríamos de alterar com *top* (cima), *right* (direita), *bottom* (baixo) ou *left* (esquerda), ficando da seguinte forma:

```
#container {
	border-top: 3px dashed #900;
	border-right: 2px dotted #CCC;
	border-bottom: 3px dashed #900;
	border-left: 2px dotted #CCC;
}
```

8.8 Formatação de textos

As propriedades **CSS** para formatação de textos são usualmente aplicadas a seletores das *tags* de cabeçalho, de parágrafo, ou com **class** e **id** específicas, como veremos. Veja uma tabela prática com as principais funções para formatação de textos:

Propriedade	Função	Valores
Font-family	Alterar o tipo da letra	Família de fontes utilizada
Font-size	Tamanho da letra	Valor numérico
Font-style	Define valores como Itálico	Italic
Font-weight	Define valores como Negrito	Bold para negrito, normal para sem negrito
Color	Cor da letra	Código hexadecimal da cor ou nome da cor em inglês
Text-align	Alinhamento de texto	Center, justify, left ou right
Text-decoration	Define valores como Sublinhado	Underline para sublinhado, none para remover sublinhado, como no caso de links

Nota: *Estas são as propriedades mais comuns para formatação de textos por CSS. Encontre a lista completa ao final da apostila.*

Exemplo:

```
h1 {
      font-family: Verdana, Geneva, sans-serif;
      font-size: 14px;
      color: #630;
      text-align: center;
      font-weight: normal;
      font-style: italic;
      text-decoration: underline;
}
```

8.9 Listas

Existem propriedades **CSS** para personalizarmos as listas ordenadas e não ordenadas. Muitas vezes em um documento *web* utilizamos listas não para o seu objetivo semântico de listar valores, mas também para funções como menus e galerias de imagens, e para isso precisaremos personalizar a lista por **CSS**.

Veja as principais propriedades e valores:

Propriedade	Função	Valores
List-style-type	Define o tipo de marcador	Geralmente usa-se none para não se ter marcadores. Para uma lista com marcadores personalizados poderemos utilizar circle (círculo), square (quadrado), disc (circulo preenchido), entre outros
List-style-position	Define a posição do marcador	inside ou outside
List-style-image	Define uma imagem como marcador	url(“imagem.jpg”)

8.10 Propriedades para tamanho e posicionamento

Ao se trabalhar com div's, imagens ou elementos, temos casos da necessidade de se alterar o tamanho do objeto ou sua posição na página. Para isto, utiliza-se as seguintes propriedades **CSS**:

Propriedade	Função	Valores
Width	Altera a largura	Valor numérico
Height	Altera a altura	Valor numérico
Left	Posicionamento em relação à esquerda	Valor numérico
Top	Posicionamento em relação ao topo	Valor numérico
Float	Flutua o elemento na página	Left, right ou none

Exemplo:

```
img {
      width: 35%;
      height: 450px;
      left: 200px;
      top: 400px;
      float: right;
}
```

8.11 Propriedade display

Existem elementos que por característica são elementos de linha, e outros elementos de bloco. Por exemplo, as *tags* ****, *tags* de cabeçalho e ** ** são elementos de linha, já as *tags* ** **, **
** são elementos de bloco, ou ainda temos casos de exibição por dados tabulares, como no caso de tabelas, pela forma em que elas exibem as informações. O atributo *display* servirá para transformar essa forma de exibição. Por exemplo, podemos fazer com que uma lista ordenada seja exibida em linha, ou um parágrafo em bloco. Veja os valores para o atributo *display*:

Exemplo:

```
ul {
      display:inline;
}
```

8.12 Propriedade overflow

A propriedade **overflow:hidden;** ocultará todos os elementos que estiverem além de seus limites, como barras de rolagem, textos, imagens ou qualquer elemento que ir além dos limites estabelecidos. Exemplo:

```
textarea {
      overflow:hidden;
}
```

8.13 Propriedade Cursor

Através da propriedade **cursor** poderemos personalizar o cursor do mouse do usuário, como se tornará ao passar sobre o elemento onde a propriedade está inserida. Possui diversos valores, mas o mais comumente utilizada é o **pointer**, para que o ponteiro do mouse se torne em uma mão indicadora, muito interessante para o caso de botões de formulários, e **default** para que o cursor não mude, como no caso de links. Poderemos também transformar o cursor em cursor de ajuda, ou de redimensionamento, esses apenas em casos especiais. Veja o exemplo:

```
input {
      cursor:pointer;
}
```

8.14 Comentários

Fazer um comentário por **CSS** é muito simples. Ele será iniciado por uma barra seguida de um asterisco e terminado de forma invertida:

```
/* Corpo do site */
body {
      margin: 3px; /* Definição das margens do documento */
}
```

Os comentários podem ser inseridos em qualquer lugar do código **CSS**, desde que não atrapalhem o desenvolvimento do código, e são muito úteis para nos ajudar a entender o conteúdo, principalmente para códigos extensos. O navegador sempre ignorará os comentários.

8.15 Pseudo-elementos

Pseudo-elementos são elementos atingidos por **CSS**, mas não por completo. Por exemplo, podemos atingir apenas a primeira palavra ou letra, preservando o resto do texto com a formatação original. Em sua sintaxe define-se o elemento a ser atingido, seguido de dois-pontos, e em seguida a **pseudo-classe** que gostaríamos de aplicar. Geralmente utiliza-se **pseudo-elementos** para a estilização de cada momento do *link*, veja:

- Link

Atinge o *link* em seu estado normal na página, antes de qualquer interação do usuário.

```
a:link {
      color: #03C;
      text-decoration: none;
}
```

- Visited

Atinge o *link* em seu estado de *link* já interagido anteriormente. Por padrão, o **XHTML** aplica uma formatação roxa a este estado, mas é mais estético que se altere isso.

```
a:visited {
      color: #900;
      text-decoration: none;
}
```

- Hover

Momento do *link* em que o usuário posiciona o mouse sobre ele. Atualmente costuma-se utilizar uma formatação de sublinhado somente para este momento.

```
a:hover {
      color: #03C;
      text-decoration: underline;
}
```

- Active

Link ativo, ou seja, o *link* selecionado ou no momento do clique do usuário.

```
a:active {
      color: #39F;
      text-decoration: none;
}
```

Obs.: *Pode-se definir qualquer formatação a cada momento do link, seja de texto para links escritos ou imagens, estas são apenas algumas sugestões.*

Os **pseudo-elementos** podem ser utilizados para qualquer *tag*, por exemplo, a interação de um usuário com uma imagem, lista ou div.

8.16 Id e class

Muitas vezes precisaremos identificar no código **XHTML** alguns elementos para assim podermos atingi-los pelas **CSS**. Identificamos os elementos de duas formas diferentes:

8.16.1 Id

Podemos dar o atributo **id** para qualquer *tag* **XHTML** por ser um atributo global. O valor do atributo deve ser uma identificação única no documento. Para evitar contratempos, utilize caracteres de A-Z, ou traços e *underline*. Exemplo:

```
<h1 id="texto">Texto formatado</h1>
```

Uma vez identificado o elemento, por padrão mundial iremos atingi-lo via **CSS** através de uma cerquilha seguida da identificação dada ao elemento no atributo **id**:

```
#texto {
      color: #900;
      font-weight: normal;
}
```

As **id's** são utilizadas quando queremos estilizar um elemento único do código **XHTML**.

8.16.2 Class

Agora imagine que precisássemos estilizar uma galeria de 50 imagens do site, colocando bordas e tamanho padrão para todas elas. Não poderíamos atingir a *tag* **<img />** sob risco de estilizar imagens que não gostaríamos também, como uma logo. Também não seria correto utilizar a mesma **id** para todas elas, por não ser semântico. Então neste caso utilizaremos **class**. Veja, usa-se **id** em uma *tag* **XHTML** para atingir um único elemento, e **class** para um grupo de elementos semelhantes. Observe como ficaria o código **XHTML** seguida da estilização das **class** por **CSS**:

```
<img src="imagem1.jpg" class="galeria" />
<img src="imagem2.jpg" class="galeria" />
<img src="imagem3.jpg" class="galeria" />
```

```
<img src="imagem4.jpg" class="galeria" />
<img src="imagem5.jpg" class="galeria" />
...
```

Atribuindo a mesma **class** a todos os diferentes elementos **XHTML**, atingiremos a todos eles pelas CSS ao mesmo tempo. Fazemos isto através de um ponto seguido do nome da **class** que gostaríamos de estilizar:

```
.galeria {
	border-width: 2px;
	border-style: ridge;
	border-color: #900;
	padding: 2px;
	width: 100px;
	height: 100px;
}
```

Pronto, todas as imagens terão a mesma formatação, e o melhor, caso queiramos alterar alguma formatação para a galeria toda, como a cor da borda por exemplo, faremos isto alterando apenas uma única linha de código.

Agora digamos que queiramos que determinada **class** funcione somente quando se refere a determinada *tag*, por exemplo um texto sendo formatado por uma **class**, mas somente quando aparecer dentro da *tag* ** **. Supondo que no código **XHTML** temos a seguinte formação: ** Texto exemplo **, o código **CSS** ficaria assim:

```
p.texto {
	font-family: Arial, Helvetica, sans-serif;
	font-size: 16px;
}
```

Caso haja a mesma **class** em outras *tags* ela não receberá a formatação, somente quando ocorrer em *tag* ** **.

8.17 Trabalhando com div's

Div's são objetos flutuantes inseridos no documento para criar divisões onde poderemos inserir qualquer conteúdo. É um elemento isento de semântica, pois pode

ser aplicado para qualquer finalidade. A div é inserida através da *tag* ** **. Como vimos em capítulos anteriores, poderemos estruturar todo o *layout* de uma página através das div's, mas como hoje podemos fazer esta função por **HTML5**, ela é mais utilizada para blocos de conteúdo específicos no *layout* e não tanto para estrutura. As div's sem **CSS** não são exibidas, nem tem alguma utilidade em si além de separar os elementos, são as **CSS** que irão desenvolvê-las. O que se faz então é dar sempre uma **id** ou **class** para elas, e então configuramos as mesmas nas folhas de estilo. As propriedades **CSS** atribuídas as div's são aquelas que já aprendemos, como estilizar seu fundo, posição, tamanho e estilos de texto. As novidades serão as propriedades **position** e **z-index**:

Propriedade	Função	Valores
Position	Define a referência para o posicionamento	Relative, caso a posição seja relativa a outra tag div relativa em que a div está inserida. Absolute, onde a referência é a tela do usuário. Fixed, a div não se movimenta com a barra de rolagem
Z-index	Define a ordem do posicionamento das div's	Valor numérico. Quanto maior a escala de valores, mais à frente a div ficará

Para definir o tamanho da div, utilizaremos as propriedades *width* e *height*. Com relação ao posicionamento, daremos as propriedades *top* e *left*, com os valores de relação de distância entre o topo e a esquerda da referência definida. A referência irá depender da opção definida na propriedade **position**.

Nota: *O uso de CSS também é chamado de tableless (menos tabelas) justamente por ele trazer uma forma mais dinâmica para estruturação de layouts, deixando tabelas apenas para dados tabulares e assim seguindo uma semântica mais apropriada.*

Veja no exemplo a estilização através de um arquivo **CSS** da página **XHTML** criada em capítulos anteriores, quando criamos um *layout* por div's (arquivo estrutura_div.html):

```
* {
      margin: 0;
      padding: 0;
      border: 0;
}
body{
```

```
	font: 12px Arial;
	background-color: #CCC;
}
#container {
	margin: 0 auto;
	width: 1000px;
}
#header {
	background: url(imagens/fundo.jpg) repeat-x;
	height: 200px;
}
#navbar {
	height: 50px;
	background-color: #666;
}
	#navbar ul {
		list-style-type:none;
		margin:0 auto;
		overflow:hidden;
		width:900px;
}
		#navbar ul li {
			background:#CCC;
			float:left;
			margin:0 10px;
			text-align:center;
			width:200px;
}
			#navbar ul li a{
				color:#FFF;
				display:block;
				font-size: 20px;
				padding:12px 0 ;
}
```

```
                #navbar ul li a:hover,
                #navbar ul li a:active{
                        background:#900;
                        color:#FFF;
}
#content {
      overflow: auto;
      background-color: #000;
}
#esquerda {
      float: left;
      width: 300px;
      height: 600px;
      background-color: #900;
      margin: 10px;
}
#centro {
      float: left;
      width: 300px;
      height: 600px;
      background-color: #900;
      margin: 10px;
      margin-left: 30px;
}
#direita {
      float: right;
      width: 300px;
      height: 600px;
      background-color: #900;
      margin: 10px;
}
#footer {
      width: 1000px;
      height: 50px;
```

```
      clear: both;
      background-color: #666;
}
p, h1, h2, h3, h4, a {
      font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;
      font-weight: normal;
      text-decoration: none;
      padding: 10px;
}
#content p {
      font-size: 14px;
}
#content a {
      float: right;
}
h1 {
      font-size: 26px;
      color: #FFF;
      text-align: center;
}
h2 {
      font-size: 20px;
      color: #FFF;
}
h3 {
      font-size: 16px;
      color: #FFF;
      text-align: center;
}
h4 {
      font-size: 12px;
      color: #036;
}
a:link {
```

```
	color: #03C;
}
a:visited {
	color: #FFF;
}
a:hover {
	color: #000;
	text-decoration: underline;
}
a:active {
	color: #099;
}
```

Observe a imagem de fundo na **div header** com **repeat-x**. Esta é uma técnica muito utilizada e que deixa o site mais muito mais leve. Se baseia em criarmos uma imagem de 1 pixel de largura e 200 de altura, por exemplo, que pode ser um degradê ou um fundo específico, e então pedimos para que ela se repita até o fim da tela, dando o efeito para toda a parte superior dela. Também poderemos utilizar a mesma técnica para a vertical, mas neste caso faremos a imagem com pouca altura, e daremos o valor **repeat-y**.


ELABORATA
INFORMÁTICA
CAPÍTULO 9
TRABALHANDO COM CSS3

WEBSTANDARDS COM HTML5

9.1 Trabalhando com CSS3

Temos uma nova versão para folhas de estilo que trazem efeitos e funcionalidades passivas de serem utilizadas nos navegadores mais modernos. Através de **CSS3** poderemos colocar sombras, arredondamentos e até animações. Muitos efeitos funcionarão em todos os navegadores modernos, já alguns apenas para navegadores específicos, outros teremos que criar um código para adaptar o efeito para os demais navegadores. Para uma lista completa de suporte de cada navegador, confira a página: http://www.findmebyip.com/litmus. Vejamos alguns efeitos possíveis de serem feitos por **CSS3**:

Nota: *Os efeitos CSS3 não são suportados por navegadores como o Internet Explorer 8.*

9.2 Sombreamento de caixas

Através de **CSS3** podemos fazer efeitos como sombra. São geralmente aplicados a *tags* div ou elementos em bloco do código **XHTML**. Para isso, precisaremos definir a cor da sombra, seu deslocamento com relação ao objeto, e o enevoamento do efeito, além de fazer códigos adaptáveis para outros navegadores. Veja:

```
#exemplo {
      width: 300px;
      height: 300px;
      box-shadow: 2px 2px 3px #CCC;
      -moz-box-shadow: 2px 2px 3px #CCC;
      -webkit-box-shadow: 2px 2px 3px #CCC;
      -o-box-shadow: 2px 2px 3px #CCC;
}
```

O primeiro valor do atributo **box-shadow** que define a cor em caixas define o deslocamento da sombra em relação ao topo, o segundo valor define o deslocamento da sombra em relação à esquerda, o terceiro valor define o enevoamento do efeito e o ultimo a cor. Observe que fizemos códigos para adaptar o mesmo efeito em cada navegador. A primeira linha para **Internet Explorer 9**, a segunda para **Mozilla Firefox**, a terceira (*webkit*) para **Google Chrome** e **Apple Safari** e a última linha para o **Ópera**.

9.3 Sombreamento de textos

Um efeito de sombreamento semelhante ao **box-shadow**, mas aplicado a textos é o **text-shadow**. Veja sua aplicação:

```
p.exemplo1 {
      text-shadow: 2px 2px 3px #CCC;
}
```

O padrão aplicado a **box-shadow** se mantém aqui, ou seja, o primeiro e o segundo valores são para o posicionamento da sombra em relação ao topo e à esquerda respectivamente, o terceiro valor para enevoamento e por último a cor desejada. Neste caso não faremos adaptação aos navegadores. Este efeito não funcionará em nenhuma versão do **Internet Explorer**.

9.4 Cantos arredondados

Um efeito inovador do **CSS3** é a possibilidade de arredondarmos os cantos de div's ou elementos de bloco, isso através do atributo **border-radius**:

```
#container {
      width: 300px;
      height: 300px;
      border-radius: 3px;
      moz-border-radius: 3px;
      webkit-border-radius: 3px;
      o-border-radius: 3px;
}
```

No exemplo temos a definição de 3 pixels de arredondamento, e a adaptação deste código aos demais navegadores modernos.

9.5 Transition

Cria um efeito de transição para o elemento. Por exemplo, a animação entre o elemento em seu estado normal, e a transformação que deve ocorrer quando posicionado o mouse sobre ele. Faremos isto através da propriedade **transition**.

```
a {
      font-family: “Trebuchet MS”, Arial, Helvetica, sans-serif;
      font-size: 14px;
      color: #F00;
      text-decoration: none;
      transition: all 3s ease;
      -moz-transition: all 3s ease;
      -webkit-transition: all 3s ease;
      -o-transition: all 3s ease;
}
a:hover {
      font-family: Tahoma, Geneva, sans-serif;
      font-size: 32px;
      color: #900;
      text-decoration: underline;
}
```

No primeiro valor de **transition** escolhemos aonde queremos aplicar a transição. Pode ser definida como **color**, para alterar apenas a cor, ou **font-size** para o tamanho, ou até **opacity** que veremos a seguir para transparência. No caso definimos **all** para todos os elementos participarem do efeito de transição. Em um segundo momento, definimos o tempo do efeito em segundos. E por último, o tipo de efeito, que possui diversas variações como **ease**, **ease-in**, **ease-out**, **linear**, entre outros. Este efeito não funcionará em nenhuma versão do **Internet Explorer**.

9.6 Transparência

Outro efeito muito interessante que o **CSS3** nos proporciona é transparência nos elementos, utilizada principalmente para div's e imagens. O efeito acontecerá através da declaração da propriedade **opacity**:

```
#container {
      width: 300px;
      height: 300px;
      background-color: #CCC;
      opacity: 0.4;
```

```
      filter: alpha(opacity=40); /* Para Internet Explorer 8 ou anterior */
}
```

Simples assim. Teremos uma escala de transparência que varia entre 0 para totalmente transparente e 1 para totalmente sólido, e a variação dos valores entre 0 e 1 para escolha da transparência desejada. Não há a necessidade de adaptar o código para os demais navegadores. Caso a versão do **Internet Explorer** seja anterior à **9**, o navegador utilizará a linha abaixo comentada, escrita em código **CSS** padrão.

9.7 @Font-face e API Google Webfonts

Um problema que os *web designers* sempre enfrentaram é a obrigação de se utilizar fontes-padrão para a criação dos seus projetos, limitando a estética dos mesmos. Através da propriedade **@font-face** da **CSS3** poderemos utilizar qualquer fonte em nossos projetos, desde que hospedemos a fonte junto com o site no servidor, e ela será automaticamente baixada quando o usuário acessar a página para que ele possa visualizar o conteúdo. Para isso, digite a seguinte configuração em uma seção a parte dentro do código **CSS,** por exemplo, no início:

```
@font-face {
      font-family: minhasfontes;
      src: url('Arista.ttf'),
      src: url('Chicago.ttf');
}
```

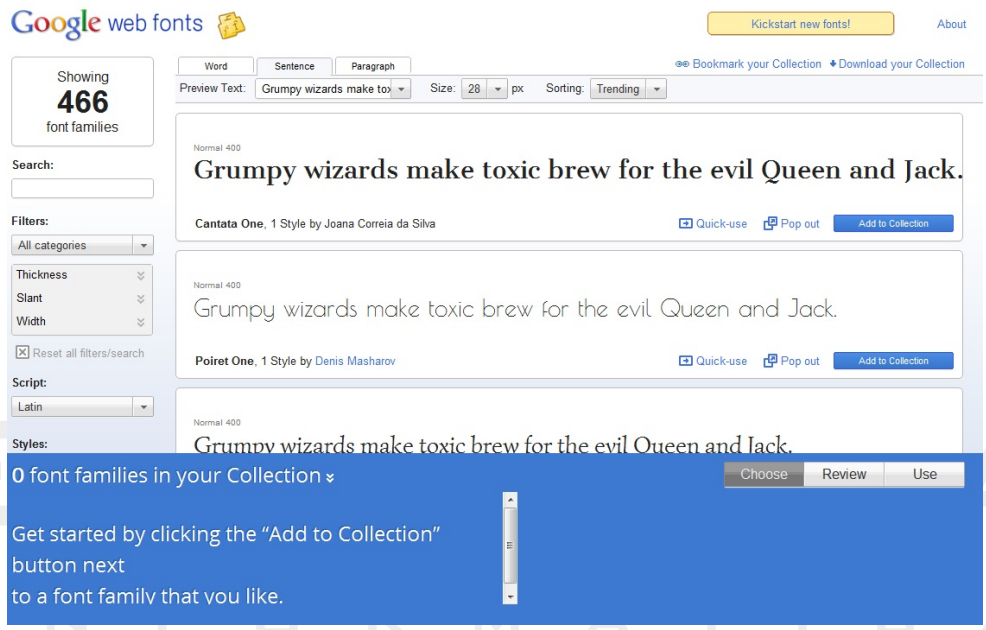
As fontes são linkadas em família na ordem dada, separadas por vírgula e entre aspas simples. O efeito é suportado em todos os navegadores e as fontes devem ter a extensão **.ttf** ou **.otf**. Fontes criadas pelo usuário com a extensão **.eot** são suportadas apenas pelo **Internet Explorer 9**. No momento em que desejar utilizar a fonte específica em um seletor, basta definir o nome dado em **@font-face** como valor da família da fonte:

```
h1 {
      font-family: minhasfontes;
}
```

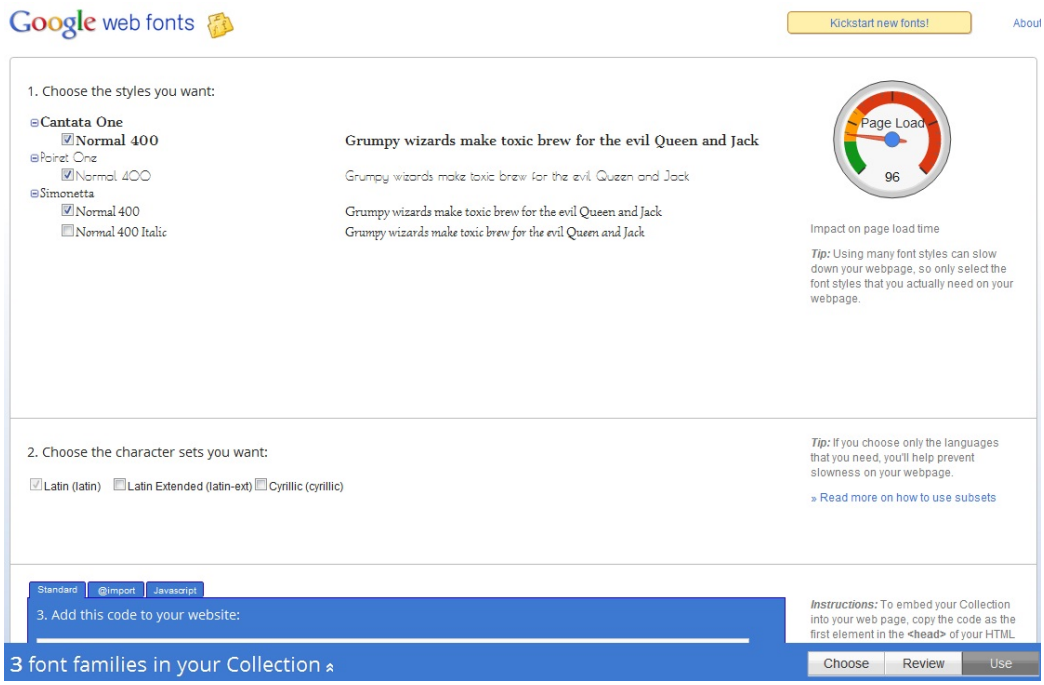
Não esqueça de colocar a fonte dentro da pasta do projeto e indicar o caminho até ela corretamente em **@font-face**, e de hospedá-la no servidor juntamente com os

demais arquivos do projeto, para que o texto possa ser aberto corretamente com a fonte desejada.

Outra forma de inserir todo o tipo de fonte em uma página é utilizando a **API** de fontes do **Google**, uma galeria imensa de fontes para web onde não há a necessidade de se fazer o download da fonte para incorporar ao documento, basta escolher a família desejada e incorporar o link ao código. Para isso, acesse a página http://www.google.com/webfonts, clique em *Start choosing fonts*:



No painel a esquerda pode-se procurar uma fonte específica pelo nome e utilizar filtros de busca para categorias e tipografia. Nas guias no topo podemos alterar a forma em que o exemplo da fonte é expresso. Ao encontrar as fontes desejadas, basta clicar em *Add to Colection*. A cada fonte escolhida, vai-se criando uma família de fontes personalizada. Ao término, pode-se clicar em *Review* na parte inferior para que possamos revisar como ficariam as fontes escolhidas em textos de exemplo. Para finalizar a escolha e incorporar no código-fonte, clicamos em *Use*:



Neste passo devemos escolher os tipos de fontes a serem incorporadas, se gostaríamos ou não de alguma variação, o tipo de caracteres, e em seguida é gerado um código para ser inserido na seção **<head></head>** do documento para que as fontes sejam linkadas a ele.

```
<link     href='http://fonts.googleapis.com/css?family=Cantata+One|Poiret+One|
Simonetta' rel='stylesheet' type='text/css'>
```

Uma vez linkado ao documento, quando quisermos utilizar a fonte basta definir seu nome na formatação por **CSS** padrão do documento, como fazemos com qualquer outra fonte. Veja o exemplo:

```
h1 {
      font-family: “Cantata One”, “Poiret One”, “Simonetta”, serif;
}
```

9.8 Gradiente

Podemos com **CSS3** criar gradientes apenas utilizando códigos. Para isso, definiremos o gradiente desejado na já conhecida propriedade **background-image**:

```
div {
      background-image: url(imagens/efeito.png); /* Para Internet Explorer */
      background-image: -moz-linear-gradient (#900, #FFF);
```

```
      background-image: -webkit-linear-gradient (#900, #FFF);
      background-image: -o-linear-gradient (#900, #FFF);
}
```

O efeito não funcionará no **Internet Explorer**. Para esses casos, mantenha uma imagem com o gradiente como no código comentado. Você também pode utilizar uma ferramenta *web* para facilitar a criação desde gradiente em: http://www.gradients.glrzad.com.

CAPÍTULO 10

ÁUDIO E VÍDEO

WEBSTANDARDS COM HTML5

10.1 A revolução do HTML5

Das muitas tecnologias importantes e inovadoras que o **HTML5** trouxe, como um novo **DTD**, uma nova forma de estruturar *layouts*, globalização de atributos e atributos novos, um dos itens mais esperados e prometidos para **HTML5** é uma inovação no trabalho de inserção de vídeo e áudio no projeto. Antes, para efetuar esta tarefa eram necessários códigos **Java Script**, adaptações de código e incorporação de elementos, ou então o trabalho com flash. Com o **HTML5** poderemos inserir estes elementos apenas inserindo *tags* **HTML** para isto, como fazemos com imagens ou qualquer outro elemento.

Nota: *Áudio e vídeo por HTML5 não são suportados por navegadores como o Internet Explorer 8.*

10.2 Áudio

Para a inserção de áudio em seu site por **HTML5** utilizamos a *tag* **<audio></audio>**:

```
<audio src="som.mp3"></audio>
```

A *tag* **<audio></audio>** pode receber os seguintes atributos:

Atributo da tag <audio></audio>	Função
src	Define a música a ser tocada
autoplay	Atributo *booleano*. Define que a música será tocada assim que estiver pronta
controls	Atributo *booleano*. Controles serão mostrados
loop	Atributo *booleano*. A música tocará novamente assim que terminar
preload	Atributo *booleano*. O áudio será primeiro totalmente carregado para depois ser tocado. É cancelado com atributo autoplay

Veja um exemplo:

```
<audio src="som.mp3" autoplay loop controls></audio>
```

Caso queiramos inserir formatos alternativos para o som, para não acontecer que tenhamos problemas de suporte de leitura do mesmo, podemos utilizar a *tag* **<source />** dentro da *tag* **<audio></audio>**:

```
<audio autoplay controls>
      <source src=”som.mp3” type=”audio/mpeg” />
      <source src=”som.wav” type=”audio/wave” />
      <source src=”som.ogg” type=”audio/ogg” />
</audio>
```

A *tag* **<source />** possui os seguintes atributos e valores:

Atributo da tag <source />	Valor
src	Define a música a ser tocada
type	Define o tipo do arquivo
codecs	Define o *codec* necessário para que a música seja tocada
media	Padrão all ou screen para o tipo de mídia

O tipo de *player* utilizado dependerá do navegador em que o site é aberto. É possível personalizá-lo através do uso de **JavaScript**.

10.2.1 Extensões suportadas

Extensão	Suporte
Arquivos .mp3	Apple Safari, Google Chrome e Internet Explorer 9
Arquivos .wav	Apple Safari, Mozilla Firefox, Google Chrome e Ópera
Arquivos .ogg	Mozilla Firefox, Google Chrome e Ópera
Arquivos .aac	Apple Safari, Google Chrome e Internet Explorer 9

10.3 Vídeo

Para inserirmos vídeos por **HTML5** utilizaremos a *tag* **<video></video>**:

```
<video src=”video.mp4”></video>
```

As extensões da *tag* **<video></video>** são semelhantes à da *tag* **<audio></audio>**, em suas propriedades e valores, assim como o uso da *tag* filha **<source />**:

```
<video autoplay controls>
```

```
      <source src="video.mp4" type="video/mp4" />
      <source src="video.ogv" type="video/ogg" />
      <source src="video.webm" type="video/webm" />
</video>
```

É possível utilizar os atributos da *tag* **<audio></audio>** vistos anteriormente e ainda outros atributos na *tag* **<video></video>** para alguns controles adicionais:

Atributo da tag <video></video>	Valor
poster	Nome e extensão de uma imagem para ser definida como capa do vídeo enquanto ele ainda estiver pausado ou carregando
audio	Valor *muted* para que o vídeo seja isento de áudio
width	Valor numérico para largura do *player*. Caso as dimensões sejam diferentes do *aspect ratio* do vídeo, o mesmo será preservado
height	Valor numérico para altura do *player*. Caso as dimensões sejam diferentes do *aspect ratio* do vídeo, o mesmo será preservado

O tipo de *player* utilizado dependerá do navegador em que o site é aberto. É possível personalizar controles através do uso de **JavaScript**.

10.3.1Extensões suportadas

Extensão	Suporte
Arquivos .mp4	Apple Safari, Google Chrome e Internet Explorer 9
Arquivos .webm	Mozilla Firefox, Google Chrome e Ópera
Arquivos .ogv	Mozilla Firefox, Google Chrome e Ópera

Veja um exemplo de um áudio e um vídeo incorporado a um site pelas novas *tags* **HTML5**:

```
<!DOCTYPE html>
<html lang="pt-br">
      <head>
            <meta charset="UTF-8" />
            <title>Áudio e Vídeo</title>
      </head>
      <body>
```

```
      <header>
            <audio preload loop controls>
                  <source src="som.mp3" type="audio/mpeg" />
                  <source src="som.wav" type="audio/wave" />
                  <source src="som.ogg" type="audio/ogg" />
            </audio>
      </header>
      <section>
            <article>
                  <video autoplay controls width="500" height="350">
                        <source src="video.mp4" type="video/mp4" />
                        <source src="video.ogv" type="video/ogg" />
                        <source src="video.webm" type="video/webm" />
                  </video>
            </article>
      </section>
      </body>
</html>
```

CAPÍTULO 11

LISTAS DE TAGS XHTML E ATRIBUTOS CSS

WEBSTANDARDS COM HTML5

11.1 Tags HTML

Lista das *tags* **XHTML** em: http://www.w3schools.com/html5/html5_ref_dtd.asp

11.2 Lista de novas tags HTML5

Segue uma lista baseada nos novos elementos **HTML5.** Muitos ainda não tem suporte em nenhum navegador atual.

Tag	Função
<article>	Cria um campo para artigo
<aside>	Conteúdo relacionado ao artigo ou para criação de sidebar
<audio>	Áudio em um documento HTML
<bdi>	Parte de um texto que pode ser formatado em diferentes direções
<canvas>	Criação de gráficos
<command>	Comando que o usuário pode invocar
<datalist>	Autocompletar em formulários como exemplos para o preenchimento do campo. Atualmente funciona apenas no Ópera e no Mozzila Firefox
<datatemplate>	Template de dados
<details>	Informações adicionais que o usuário pode visualizar ou esconder
<device>	Permite ao usuário dar o acesso à página para um dispositivo
<embed>	Aplicação externa com conteúdo interativo
<figcaption>	Define uma legenda para um elemento <figure>
<figure>	Ilustrações, imagens, fotos, associados com alguma legenda
<footer>	Cria uma seção na página que pode ser utilizada para criação do Rodapé.
<header>	Cria uma seção na página que pode ser utilizada para criação do Topo.
<hgroup>	Container para elementos de título do nível h1 ao h6
<keygen>	Chave de acesso
<mark>	Texto destacado para fins de referência
<meter>	Medição dentro de um intervalo predefinido
<nav>	Lista de links para navegação
<output>	Resultado de um cálculo
<progress>	Progresso de uma tarefa como a execução de um script em Java
<rt>	Componente do texto em uma anotação
<section>	Seção dentro de um artigo

<source>	Exibe múltiplos recursos multimídia
<time>	Define data e hora de uma publicação por exemplo. Os navegadores atuais ainda não oferecem suporte para seu funcionamento
<track>	Legenda para elementos multimídia
<video>	Vídeo em um documento html
<wbr>	Possível quebra de linha

11.3 Atributos Globais HTML5

Atributos globais são atributos que podem ser inseridos em qualquer *tag*. Com **HTML5** muitos atributos que eram específicos se tornaram globais e outros foram incorporados. Muitos deles ainda não tem suporte em nenhum navegador atual. Veja a lista:

Atributo	Função
accesskey	Especifica uma tecla de atalho para dar foco ao elemento
class	Atribui um nome de classe para ser estilizado por folhas de estilo
contenteditable	Especifica se o elemento é editável
contextmenu	Atribui um menu de contexto para aparecer quando o usuário utilizar o botão auxiliar do mouse
dir	Direção do texto
draggable	Especifica se o elemento é passível de ser arrastado
dropzone	Especifica se o elemento arrastado será copiado, movido ou linkado
hidden	Esconde o elemento
id	Atribui uma identificação ao elemento para ser estilizado por folhas de estilo
lang	Especifica o idioma
spellcheck	Atribui um corretor ortográfico
style	Para estilizações inline
tabindex	Especifica a ordem de acesso pela tecla tab
title	Texto ao posicionar o ponteiro do mouse sobre o elemento

11.4 Lista de propriedades CSS

Propriedade	Valores
background	Color image repeat attachment position
background-attachment	Fixed ou scroll
background-image	url(nomedaimagem.jpg)
background-color	Código hexadecimal ou nome da cor
background-position	Valor numérico seguido da unidade de medida. Primeiro valor para esquerda, segundo para topo
background-repeat	No-repeat, repeat, repeat-x ou repeat-y
border	width style color
border-color	Código hexadecimal ou nome da cor
border-style	None, dotted, dashed, solid, double, groove, ridge, inset ou outset
border-width	Valor numérico seguido da unidade de medida
border-bottom	width style color
border-bottom-color	Código hexadecimal ou nome da cor
border-bottom-style	None, dotted, dashed, solid, double, groove, ridge, inset ou outset
border-bottom-width	Valor numérico seguido da unidade de medida
border-left	width style color
border-left-color	Código hexadecimal ou nome da cor
border-left-style	None, dotted, dashed, solid, double, groove, ridge, inset ou outset
border-left-width	Valor numérico seguido da unidade de medida
border-right	width style color
border-right-color	Código hexadecimal ou nome da cor
border-right-style	None, dotted, dashed, solid, double, groove, ridge, inset ou outset
border-right-width	Valor numérico seguido da unidade de medida
border-top	width style color
border-top-color	Código hexadecimal ou nome da cor
border-top-style	None, dotted, dashed, solid, double, groove, ridge, inset ou outset
border-top-width	Valor numérico seguido da unidade de medida

clear	None, left, right ou both
color	Código hexadecimal ou nome da cor
cursor	Default, pointer, crosshair, e-resize, help, move, ne-resize, n-resize, nw-resize, se-resize, s-resize, sw-resize, text, wait ou w-resize
display	Block, inline, none, inline-block, inline-table, compact, list-item ou table
font	Style variant weight size family
font-family	Família de fontes
font-size	Valor numérico seguido da unidade de medida
font-style	Normal, italic ou oblique
font-variant	Normal ou small-caps
font-weight	Normal, bold, bolder, lighter ou valor numérico
height	Valor numérico seguido da unidade de medida
left	Valor numérico seguido da unidade de medida
letter-spacing	Valor numérico seguido da unidade de medida
line-height	Valor numérico seguido da unidade de medida
list-style	Position; imagem ou position; type
list-style-image	None ou url(nomedaimagem.jpg)
list-style-position	Outside ou inside
list-style-type	None, disc, circle, square, decimal, lower-roman, upper-roman, lower-alpha ou upper-alpha
margin	Top right bottom left
margin-left	Valor numérico seguido da unidade de medida
margin-right	Valor numérico seguido da unidade de medida
margin-bottom	Valor numérico seguido da unidade de medida
margin-top	Valor numérico seguido da unidade de medida
overflow	Visible, hidden, scroll ou auto
padding	Valor numérico seguido da unidade de medida
padding-bottom	Valor numérico seguido da unidade de medida
padding-left	Valor numérico seguido da unidade de medida
padding-right	Valor numérico seguido da unidade de medida
padding-top	Valor numérico seguido da unidade de medida
position	Absolute, relative, fixed ou static
text-align	Left, right, center ou justify
text-decoration	None, underline, overline, line-through ou blink

text-indent	Valor numérico seguido da unidade de medida
text-transform	None, capitalize, uppercase ou lowercase
top	Valor numérico seguido da unidade de medida
vertical-align	Baseline, bottom, middle, sub, super, text-bottom, text-top ou top
visibility	Hidden ou visible
width	Valor numérico seguido da unidade de medida
Word-spacing	Valor numérico seguido da unidade de medida
z-index	Valor numérico

11.5 Lista das novas propriedades CSS3

Propriedade	Valores
backface-visibility	Hidden
background-origin	Border-box, padding-box ou content-box
background-clip	Border-box, padding-box ou content-box
background-size	Valor numérico seguido da unidade de medida como primeiro valor para largura e Valor numérico seguido da unidade de medida para valor da altura
border-color	Código hexadecimal ou nome da cor
border-image	url(imagem.jpg) para primeiro valor, Valor numérico seguido da unidade de medida para segundo e terceiro valor e valor round para quarto valor
border-radius	Valor numérico seguido da unidade de medida
box-sizing	Content-box ou border-box
box-shadow	Valor numérico seguido da unidade de medida para os três primeiros valores e no quarto valor código hexadecimal ou nome da cor
column-count	Valor numérico
column-gap	Valor numérico seguido da unidade de medida
column-rule	No primeiro valor definimos um valor numérico seguido da unidade de medida, em seguida um estilo, que pode ser None, dotted, dashed, solid, double, groove, ridge, inset ou outset e por fim o Código hexadecimal ou nome da cor
opacity	Valor numérico entre zero e 1
outline-offset	Valor numérico seguido da unidade de medida
perspective	Valor numérico

perspective-origin	Um primeiro valor numérico seguido da unidade de medida para o eixo x e um segundo Valor numérico seguido da unidade de medida para o eixo y
resize	None, both, horizontal ou vertical
text-shadow	Os primeiros dois valores são valores numéricos seguidos da unidade de medida para deslocamento horizontal e vertical respectivamente, um terceiro valor para enevoamento e o quarto valor como código hexadecimal ou nome da cor
text-overflow	Clip, ellipsis ou string
transform	None, matrix, matrix3d, translate, translate3d, translatex, translatey, translatez, scale, scale3d, scalex, scaley, scalez, rotate, rotate3d, rotatex, rotatey, rotatez, skew, skewx, skewy ou perspective
transition	Primeiro valor com all ou a definição do que animar, segundo valor em tempo e terceiro valor como ease, ease-in, ease-out, ease-in-out, linear ou cubic-bezier
word-wrap	Normal ou break-word
@keyframes	Para definições de animações
@font-face	Para definição de fontes externas


ELABORATA
INFORMÁTICA
CAPÍTULO 12
TABELAS PARA
REFERÊNCIA

12.1 Tabela de cores

Como vimos, em códigos **XHTML** e **CSS** são utilizadas cores com seu nome em inglês ou o código hexadecimal que lhe dá origem. Este código é formado pela codificação **RGB**, onde temos valores para *Red* (Vermelho), *Green* (Verde) e *Blue* (Azul). No **CSS**, quando ocorre um código de cores em que se repetem cores consecutivas, como em **#FF6600**, podemos escrever somente três caracteres, sem a repetição de valores (**#F60**). Veja a seguir uma tabela com as principais cores e a tabela completa em: http://www.elaborata.com.br/tabela_cores.html.

Cor	Código Hexadecimal	Cor	Código Hexadecimal
White	#FFFFFF	Purple	#A020F0
Black	#000000	Blue1	#0000FF
Grey	#BEBEBE	Blue2	#0000EE
LightGrey	#D3D3D3	PaleTurquoise1	#BBFFFF
Blue	#0000FF	Cyan1	#00FFFF
SkyBlue	#87CEEB	Cyan2	#00EEEE
Turquoise	#40E0D0	Green1	#00FF00
Cyan	#00FFFF	Green2	#00EE00
Aquamarine	#7FFFD4	Yellow1	#FFFF00
SeaGreen	#2E8B57	Yellow2	#EEEE00
Green	#00FF00	Gold1	#FFD700
GreenYellow	#ADFF2F	Orange1	#FFA500
LimeGreen	#32CD32	Orange2	#EE9A00
LightYellow	#FFFFE0	Red1	#FF0000
Yellow	#FFFF00	Red2	#EE0000
Gold	#FFD700	Magenta1	#FF00FF
Goldenrod	#DAA520	Magenta2	#EE00EE
Beige	#F5F5DC	Plum1	#FFBBFF
Firebrick	#B22222	grey11	#1C1C1C
Brown	#A52A2A	grey21	#363636
Salmon	#FA8072	grey31	#4F4F4F
Orange	#FFA500	grey41	#696969
DarkOrange	#FF8C00	grey51	#828282
OrangeRed	#FF4500	grey61	#9C9C9C
Red	#FF0000	grey71	#B5B5B5

Pink	#FFC0CB	gray81	#CFCFCF
Magenta	#FF00FF	gray91	#E8E8E8

12.2 Tabela de fontes-padrão

Como vimos, não poderemos utilizar todas as fontes que quisermos no nosso código **XHTML** e **CSS**. Por questão de acessibilidade utilizamos somente fontes-padrão, que são fontes que temos a certeza de que existem em todos os sistemas operacionais modernos (a não ser que se utilize @font-face do **CSS3** ou a **API** de *Webfonts* do **Google**). No caso de **CSS** utilizamos famílias de fontes, para caso a primeira fonte não exista no computador do usuário, ela seja substituída pela seguinte e assim sucessivamente para exibir a informação. Veja quais são:

Fonte-padrão	Família de fontes
Arial	Arial, Helvetica, sans-serif
Arial Black	Arial Black, Gadget, sans-serif
Comic Sans MS	Comic Sans MS, cursive
Courier New	Courier New, Courier, monospace
Georgia	Georgia, Times New Roman, Times, serif
Lucida Console	Lucida Console, Monaco, monospace
Lucida Sans Unicode	Lucida Sans Unicode, Lucida Grande, sans-serif
MS Serif	MS Serif, New York, serif
Palotino Linotype	Palotino Linotype, Book Antiqua, Palotino, serif
Tahoma	Tahoma, Geneva, sans-serif
Times New Roman	Times New Roman, Times, serif
Trebuchet MS	Trebuchet MS, Arial, Helvetica, sans-serif
Verdana	Verdana, Geneva, sans-serif

No caso das **CSS**, devemos colocar uma fonte que tenha mais de um nome em sua descrição entre aspas. Exemplo: “Trebuchet MS”, Arial, Helvetica, sans-serif.

12.3 Tabela de caracteres

Em um documento XHTML podemos escrever caracteres com acentos, cedilha e caracteres especiais, mas pode acontecer de um usuário que utilize um idioma padrão em seu sistema operacional que não contenha esses caracteres. Neste caso, os

caracteres não serão exibidos, ou serão exibidos de forma deturpada, como um caractere inválido. Para evitar este tipo de problema, por questão de acessibilidade devemos procurar digitar conteúdos com caracteres diferentes no formato de códigos que levam a ele e são interpretados pelos navegadores em qualquer idioma. São eles:

Caractere	Código	Caractere	Código	Caractere	Código	Caractere	Código
Á	Á	É	É	Ô	Ô	®	®
á	á	é	é	ô	ô	©	©
Â	Â	Ê	Ê	Õ	Õ	Ñ	Ñ
â	â	ê	ê	õ	õ	ñ	ñ
À	À	Í	Í	Ú	Ú	Ë	Ë
à	à	í	í	ú	ú	ë	ë
Ã	Ã	Ó	Ó	ç	ç	Ü	&Uuml
ã	ã	ó	ó	&	&	ü	ü

Nota: A definição do charset como utf-8 resolve o problema da exibição de caracteres para todos os idiomas derivados do latin. Se você utilizar o charset como iso-8859-1 deverá mudar os caracteres especiais para os especificados na tabela anterior.

CAPÍTULO 13

CONCLUSÃO

WEBSTANDARDS COM HTML5

13.1 Leitura complementar

No que diz respeito a semântica e padrões *web* é sempre bom conferir o site nacional da **W3C:** http://www.w3c.br;

A própria **W3C** oferece um portal na *web* para o aprendizado de novos efeitos e suportes a **HTML**, **CSS**, **HTML5** e **CSS3** em: http://www.w3schools.com.

13.2 Referências para consulta

No que diz respeito as **CSS**, o site **Maujor** é a maior referência nacional atual: http://www.maujor.com;

Lista com todas as principais *tags* **HTML** em: http://codigofonte.uol.com.br/artigo/html-xhtml/principais-tags-de-html.

13.3 Referências bibliográficas:

- Samy Silva, Maurício; **HTML5 A Linguagem de Marcação que Revolucionou a Web – Primeira Edição**; Novatec Editora, 2011.

Rua Monsenhor Celso, 256 - 1º andar - Curitiba – Paraná

41.3324.0015

41.99828.2468

Agradecimentos:

Equipe Elaborata Informática