Aula 2: Fundamentos do XHTML

Prof. Daniel Caetano

Objetivo: Introduzir a linguagem XHTML e apresentar suas principais tags.

Bibliografia: W3C, RAMALHO, 1999; BOENTE, 2006; NIELSEN, 2000.

INTRODUÇÃO

Desde sua criação até os dias atuais, a Web evoluiu muito. Grande parte desta evolução só foi possível porque a linguagem usada para descrever as páginas da Web também evoluiu.

Com o surgimento de novos recursos nesta linguagem de descrição, a maneira de utilizá-la corretamente também sofreu mudanças profundas com o passar dos anos, fazendo com que muitos profissionais ficassem desatualizados, usando técnicas há muito desaconselhadas pelo W3C, o World Wide Web Consortium, responsável pela padronização da Web.

A partir de agora, veremos a maneira "correta" de usar a linguagem de descrição de páginas Web para garantir que nossas páginas e web sites sejam flexíveis e facilmente administráveis.

1. A ORIGEM DA LINGUAGEM DE DESCRIÇÃO WEB

Conceitos Chave:

- Até 1990: SGML (Linguagem de Marcação Geral Padronizada)
  - * Documentos legíveis por humanos e por máquinas
  - * Tags: especificam elementos para a máquina
    - + Indicados por "<" e ">"
    - + Marcadores de início e fim
    - + Ex.: <QUOTE>Texto</QUOTE>
- SGML é *geral*, servindo para muitas coisas.
  - * Compreensão relativamente complexa
- Tim Berners Lee => Aplicação do SGML
  - * HTML: Linguagem de Marcação de HiperTexto.
    - + Usa "tags" como as da SGML
    - + Indicar funções etruturais de um texto:
      - = Título, subtítulo, parágrafo, endereço de e-mail...
  - * **Não usar caracteres especiais/espaços em nomes de arquivo**
  - * Arquivos com nome sempre finalizado em **.html**

A linguagem de descrição de páginas Web foi inventada no início da década de 1990, por Tim Berners Lee, baseada em uma linguagem chamada SGML (Standard Generalized Markup Language - Linguagem de Marcação Geral Padronizada). A SGML serve para criar documentos que sejam legíveis tanto por seres humanos quanto por máquinas. O método para atingir este objetivo é indicar qual a função de cada trecho de texto através de "tags", que são indicadores separados pelos caracteres "<" e ">", como por exemplo: <QUOTE>Texto</QUOTE>. Observe que existe um marcador de início e um marcador de fim, sendo que neste último um caractere "/" precede o nome do indicador.

Entretanto, como o próprio nome da SGML diz, ela é "geral", ou seja, serve para muitas coisas. Por esta razão, ela é de "compreensão" relativamente complexa tanto para humanos quanto para máquinas. Assim, quando Tim Berners Lee pensou na Web, ele fez uma *aplicação* da SGML, simplificando-a para os usos que ele tinha em mente: documentos em hipertexto. Por esta razão, Lee chamou essa linguagem de HTML: HyperText Markup Language - Linguagem de Marcação de HiperTexto.

A idéia do HTML é usar "tags" (que serão vistas em breve) para indicar qual trecho de um texto é um título, qual é um subtítulo, qual é um parágrafo, qual é um endereço de e-mail, qual é um trecho de um programa de computador, qual é uma citação, qual é um link... e assim por diante.

Um detalhe importante é que define-se que, na web, **nenhum** arquivo terá qualquer caractere especial (permitidos apenas o sublinhado "_" e o traço "-"), nem mesmo devem conter espaços ou caracteres acentuados. Um arquivo html deve sempre ter seu nome terminado por .**html**. Mais adiante serão vistos mais detalhes sobre isso.

1.1. O Formato XHTML

Com a criação do formato XML (eXtensible Markup Language) e sua grande flexibilidade, foi natural a implementação do HTML através do XML, em um formato que ganhou o nome de XHTML (eXtensible Markup Language).

Essencialmente, o XHTML é um HTML que deve atender a todas as regras do XML, ou seja, é intolerante a "erros de formação", que tornam um documento mal formado. Mas o que é um documento bem formado?

Um documento bem formado tem as seguintes características:

Todas as *tags* e parâmetros devem ser grafadas em minúsculas

Exemplo incorreto: <P CLASS="teste">Parágrafo</P>
Exemplo correto: Parágrafo

- É obrigatório o uso de *tags* de fechamento

Exemplo incorreto: Parágrafo 1
Parágrafo 2
Exemplo correto: Parágrafo 1
Parágrafo 2

- Elementos vazios também devem ser fechados

Exemplo incorreto: Quebra
Régua<hr>
<img src="imagem.gif" alt="Imagem">
Exemplo correto: Quebra
Régua<hr />

- Todas as *tags* devem ser adequadamente aninhadas

Exemplo incorreto: ênfase
Exemplo correto: ênfase

- Os valores dos atributos devem sempre estar entre aspas

Exemplo incorreto: Parágrafo
Exemplo correto: Parágrafo

Além disso, é **obrigatório** o uso do elemento raiz <html> e também a definição de um tipo de documento, com a tag <DOCTYPE>. Existem algumas mudanças adicionais, que são comentadas ao longo do texto, como a *proibição* do uso de tags de posicionamento (como) e visuais (como <font>) ou atributos como *name* e *width*. Adicionalmente, o uso do parâmetro *alt* é obrigatório na *tag* <img>.

2. A ESTRUTURA DE UM DOCUMENTO HTML

Conceitos Chave:
- Leitura automatizada => Exige estrutura de documento
  * XHTML => Tem estrutura fixa
    + Sempre se repete
- Definição do tipo de documento
  * Transitional / Frameset / Strict
- Todo o conteúdo em XHTML de uma página deve ficar na região HTML
  * <html> ... </html>
  * O que estiver fora disso pode ser ignorado pelo navegador

- Duas partes importantes:
    * Cabeçalho: <head> ... </head>
        = Informações sobre a página
        = Nada de conteúdo
    * Corpo: <body> ... </body>
        = Conteúdo apresentado pelo navegador na página em si
- Relação Cabeçalho/Corpo x Área Frame/Cliente

Computadores são bastante metódicos e, sendo assim, tudo que é feito para que um computador leia, precisa ter uma certa estrutura. Assim, antes de apresentarmos algumas das tags do XHTML, é importante apresentar a estrutura de um documento XHTML.

É importante observarmos esta estrutura com atenção - e memorizá-la, porque ela será usada com freqüência e, se a estrutura de um documento XHTML não estiver correta, corre-se o risco de o navegador não interpretá-la corretamente, apresentando-a com falhas para os usuários de nossas páginas.

A estrutura fundamental de uma página XHMTL é composta pela tag DOCTYPE e mais duas seções importantes: o cabeçalho e o corpo. Cada uma destas seções fica indicada por *tags* específicas do próprio HTML.

A primeira coisa a se fazer, é indicar no documento o tipo de documento XHTML que estamos criando, com a tag DOCTYPE. Existem 3 tipos de tipo de documento:

Transitional: usado para documentos em fase de adaptação entre HTML e XHTML;
Tag DOCTYPE para Transitional:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EM"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

Frameset: o mesmo que transitional, mas também permite o uso de frames;
Tag DOCTYPE para Frameset:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EM"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">

Strict: só permite o uso de elementos de estruturação e *tags* modernas.
Tag DOCTYPE para Strict:
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

Após isso, indicamos que estamos incluindo uma seção de código HTML. Isso é feito pelas tags **<html>** e **</html>**. Estas tags indicam que tudo que estiver entre elas deve ser processado como HTML. Por outro lado, teoricamente, tudo que estiver fora delas deve ser ignorado pelo navegador.

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        [ Trecho de Código HTML ]
</html>
```

Em seguida, deve-se indicar a região do cabeçalho, delimitada pelas tags **<head>** e **</head>**. Nesta região serão indicadas muitas informações da página que *não são apresentadas na área cliente do navegador*. Mais adiante essa seção será apresentada com mais detalhes.

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                [ Dados do Cabeçalho ]
        </head>
        [ Mais Código HTML ]
</html>
```

Finalmente, deve ser indicada a região do corpo da página, delimitada pelas tags **<body>** e **</body>**. Esta região conterá as informações que *são apresentadas na área de cliente do navegador*. Mais adiante essa seção será apresentada com mais detalhes.

Com isso, praticamente completa-se a página Web mais simples possível - que é uma página Web vazia, que terá essa cara:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
        <head>
                [ Dados do Cabeçalho ]
        </head>
        <body>
                [ Dados do Corpo da Página ]
        </body>
</html>
```

Resumindo e ressaltando as funções das seções de cabeçalho <*head*> e de corpo <*body*>, o conteúdo da seção <head> não é para o usuário, e sim para o programa navegador. Por esta razão, o conteúdo da seção <head> não é mostrado para o usuário. Já o conteúdo da seção <body> é o conteúdo que o navegador mostrará para o usuário como sendo a página Web.

Observe que tanto a seção <head> quanto a seção <body> estão inteiramente contidas entre as tags <html> e </html>; caso isso não fosse respeitado, a página poderia apresentar problemas em alguns navegadores. Esta ordenação de tags <html>, <head> e <body> é a estrutura fundamental de uma página Web e toda página será iniciada exatamente com esta estrutura.

Como a página que vamos criar terá seu conteúdo na língua portuguesa, é conveniente indicar isso em nosso documento, de forma que os navegadores e interfaces de busca saibam qual é a língua do conteúdo de nossa página. Isso por ser feito com o parêmetro modificador *xml:lang* na tag <html>, com o valor "pt-BR" (português do Brasil). A página ficará como apresentado a seguir.

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
                [ Dados do Cabeçalho ]
        </head>
        <body>
                [ Dados do Corpo da Página ]
        </body>
</html>
```

Esta indicação, dentre outras coisas, permite que serviços como o Google saiba em qual língua uma página foi criada e permite que a busca seja "filtrada" por esta característica.

No Google as opções são: "todas as páginas", "só páginas em português" e "só páginas do Brasil". Esta indicação influi na segunda opção, embora não seja o único critério que o Google usa.

Com isso fica concluída a estrutura fundamental de um documento HTML; vejamos agora detalhes sobre o cabeçalho e o corpo da página.

2.1. O Cabeçalho da Página

Conceitos Chave:

- XHTML => Cabeçalho => <head> ... </head>
  - * Navegadores buscam informações: Firefox, IE, Opera... agentes...
- Informações básicas de cabeçalho:
  - * Título da Página: <title> ... </title>
    - + Barra de título do navegador, Bookmark / Favoritos
  - * Outras informações
    - + Autor, folha de estilos, códigos javascript...

Anteriormente foram apresentadas as duas principais seções de um documento XHTML, que é usado para compor uma página Web. A primeira delas é o cabeçalho e, como foi dito anteriormente, esta seção contém informações para o navegador, que pode ser o Firefox, Internet Explorer, Opera, Chrome... mas também pode ser um navegador "spider" de algum serviço de busca como o Google.

> Um navegador "spider" é um agente web que vasculha as páginas e seus links para realizar algum tipo de tarefa. No caso dos navegadores spider do Google, sua função é rastrear e indexar páginas web.

Cada um destes navegadores busca coisas distintas neste cabeçalho, mas uma coisa que todos eles investigam é o *título da página*. Em XHTML, identificamos o título da página, dentro do cabeçalho, pelas tags <title> e </title>. Esta *tag* é considerada **obrigatória** no XHTML strict. Por exemplo:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
                <title>Título da página Web!</title>
        </head>
        <body>
                [ Dados do Corpo da Página ]
        </body>
</html>
```

Gravando este texto com o nome de "pagina.html" e abrindo-o em um navegador, você poderá observar que o texto "Título da página Web!" não aparece em lugar algum na página; entretanto, ele foi parar em um lugar especial: na barra de título do navegador.

Caso você faça um "bookmark" desta página, ou seja, coloque-a nos seus "favoritos", também será este texto entre **<title> e </title>** que irá ser indicado nos favoritos.

Existem outras indicações que podem ser colocadas no cabeçalho:

- O nome do autor da página
- Palavras chave
- Tipo de codificação de caractere da página
- Instruções para o navegador (recarga ou cache, por exemplo)
- Códigos em javascript
- ...

Uma outra informação obrigatória para o XHTML strict é a definição da codificação de caracteres. O padrão é indicar que os caracteres estão em UTF-8, mas é possível identificá-los no formato do Windows:

Padrão:

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

Windows:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
```

O resultado é:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                [ Dados do Corpo da Página ]
        </body>
</html>
```

Por enquanto vamos nos limitar à essas informações, que são as obrigatórias e mais importantes de todas as informações que podem figurar no cabeçalho.

Apenas a título de exemplo, algumas das indicações possíveis:

Palavras chave: <meta name="keywords" content="palavra1, palavra2, ..." />
Autor da página: <meta name="owner" content="Nome do Autor" />
Desligar cache: <meta http-equiv="pragma" content="no-cache" />
Ícone: <link rel="icon" href="imagem.png" type="image/png" />

2.2. O Corpo da Página

Conceitos Chave:

- XHTML => Corpo => <body> ... </body>
  - * Conteúdo apresentado ao usuário pelo navegador
  - * Parágrafo: ... é uma das tags principais do XHTML
- Nota: Em XHTML, quebras de linha (enter) são ignoradas!

O corpo da página, como foi dito anteriormente, deve ser delimitado pelas tags <body> e </body>; tudo que aparecerá na página deverá estar indicado nesta região. A função da maioria das tags do XHTML é *explicar ao navegador qual é o conteúdo da página*.

Uma vez que o principal formato de conteúdo nas páginas web é o formato texto, uma das tags mais fundamentais do XHTML é a tag de Parágrafo**: **. Como um parágrafo tem um início e um fim, então existe também uma tag de fechamento de parágrafo: ** **. Assim, no exemplo a seguir por ser verificado como é possível indicar um parágrafo da maneira correta em uma página XHTML.

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                <p>Texto da página Web!</p>
        </body>
</html>
```

Aqui convém ressaltar um fato importante: o navegador, ao interpretar o XHTML, ignora as quebras de linha. Isso significa que indicar isso:

```
<p>Texto da página Web!</p>
```

Tem o mesmo efeito de escrever:

```
<p>
        Texto da página Web!
</p>
```

Que tem o mesmo efeito de escrever:

```
<p>
        Texto da
        página Web!
</p>
```

Nas próximas seções e aulas veremos muitas outras tags que podem ser usadas em um documento XHTML.

3. TAGS ESTRUTURANTES ADICIONAIS

Conceitos Chave:

- Manchetes / Títulos / Subtítulos
  - * Maior para menor evidência
    - + h1, h2, h3, h4, h5 e h6
    - + <h1>Manchete</h1>
  - * Devem sempre ser usados
    - + Índices automatizados
    - + Navegadores para cegos
- Quebra de linha
  - * Intervenção pontual
    - +
  - * Não substitui <p>!
- Linha de separação horizontal
  - * Cria uma separação entre duas seções da página
    - + <hr />

Manchetes/Títulos/SubTítulos: O XHTML tem tags para especificar trechos de um texto que são manchetes, títulos ou, ainda, subtítulos. Na realidade, existem 6 níveis de títulos, sendo que o nível 1 tem maior evidência até o nível 6, de menor evidência. Por exemplo:

```
<h1>1. Seção h1</h1>
    <h2>1.1. Seção h2</h2>
        <h3>1.1.1. Seção h3</h3>
            <h4>1.1.1.1. Seção h4</h4>
                <h5>1.1.1.1.1. Seção h5</h5>
                    <h6>1.1.1.1.1.1. Seção h6</h6>
```

Isso apareceria na tela mais ou menos assim:

1. Seção h1
1.1. Seção h2
1.1.1. Seção h3
1.1.1.1. Seção h4
1.1.1.1.1. Seção h5
1.1.1.1.1.1. Seção h6

Estas tags devem ser usadas com sabedoria para indicar as diversas seções do texto da página. Bons navegadores poderão, no futuro, fazer índices automáticos com estas seções, além de manter uma compatibilidade com os já existentes navegadores para cegos, que lêem estes títulos para que a pessoa escolha qual das seções quer ouvir primeiro.

Quebra de Linha: existe uma tag no XHTML usada apenas em casos muito especiais em que se deseja forçar a quebra de uma linha em uma determinada posição. Esta tag é definida por **
**, de Break Row ou line BReak (quebra de linha, em inglês). Por se tratar de uma intervenção pontual, ela deve ser sempre fechada com a / antes do >: **
**... é incorreto escrever **
. Convém lembrar, também, que **é incorreto o uso da quebra de linha em substituição à tag de parágrafo.

** Este parágrafo será quebrado no meio, bem aqui
e continua depois **

O texto acima será apresentado da seguinte forma:

Este parágrafo será quebrado no meio, bem aqui
e continua depois

Linha de Separação: Um recurso muito usado para tornar claro quando uma seção termina e onde outra começa costuma ser a linha de separação horizontal. O HTML tem uma tag para apresentar este tipo de separador: **<hr />**, de Horizontal Ruler (Régua Horizontal em inglês). Exemplo de uso:

```
<h1>1. Seção h1</h1>
        <h2>1.1. Seção h2</h2>
<hr />
<h1>2. Seção h1</h1>
        <h2>2.1. Seção h2</h2>
```

Será apresentado da seguinte forma:

1. Seção h1
1.1. Seção h2

---

2. Seção h1
2.1. Seção h2

4. CODIFICAÇÃO DE ACENTUAÇÃO

Conceitos Chave:

- Uso de acentos
  - * E computadores que não reconhecem acentos?
  - * Por que não reconhecem?
    - + ASCII
    - + ASCII Estendido
- Soluções
  - * Indicar qual codificação está sendo usada
  - * Usar tags de acentuação XHTML
- Indicação de Codificação no Cabeçalho
  - * Tag <meta>
    - + Modificador http-equiv="Content-Type"
    - + Modificador content="..."
      - = Codificação: iso-8859-1
      - = Codificação: utf-8
- Tags de Acentuação HTML
  - * Por que usar?
  - * &__nome;
    - + á + &Àgrave;
    - + ô + ü
    - + õ + ç

Se, enquanto criamos uma página, simplesmente escrevermos um texto acentuado, sem maiores preocupações, estaremos criando uma página incompatível com o sistema de muitos usuários.

Basicamente, a razão para isso acontecer é que usamos **acentuação** no texto e, em muitos países, a língua utilizada não tem acentuação. Sendo assim, os computadores e navegadores daqueles países simplesmente não entendem o que esta acentuação significa, se não forem especialmente instruídos para isso.

Para explicar porque isso ocorre, é preciso voltar um pouco às origens dos sistemas computacionais. Como você já deve saber, todo texto digitado no computador é convertido em números, quando armazenado na memória ou em um arquivo. Cada número representa uma letra. Assim, se o número 65 representa a letra A, 66 representará a letra B, 67 representará a letra C... e assim por diante. Esta codificação é chamada "Codificação ASCII", criada por uma associação norte-americana.

Esta codificação acabou se tornando padrão mundial, mas ela só definia 128 caracteres básicos (números 0 a 127) e, como na língua inglesa não há acentos ou cedilha, estes não fazem parte da padronização. Sendo assim, cada país acabou desenvolvendo um complemento de 128 caracteres (números de 128 a 255), usando estes números adicionais para indicar seus caracteres acentuados.

No Brasil, por exemplo, o caractere "á" é representado pelo número 225 e o caractere "Ã" é representado pelo número 195. Ocorre que, como cada país fez a sua extensão, estes caracteres e seus números não são padronizados internacionalmente, e mesmo dentro de um país, em alguns casos, houve diversas versões destas codificações.

Assim, se escrevemos um "à" em um arquivo aqui no Brasil (código 224) e o abrirmos em um navegador na Rússia, ao invés de um "à" veremos um "R ao contrário", já que na Rússia o código 224 foi usado para essa letra específica do alfabeto cirílico (russo).

Apenas recentemente foi criada uma codificação internacional, chamada **UNICODE**, que existe em três variações: UTF-8, UTF-16 e UTF-32, variando o número de caracteres disponível para cada uma delas. UTF-8 tem 256 caracteres e é suficiente para a maioria das línguas ocidentais. UTF-16 possui 16384 caracteres e possui caracteres de todas as línguas do mundo, mas possui apenas um conjunto limitado de caracteres de línguas como japones e chinês. O UTF-32 tem mais de 4 bilhões de caracteres e possui representação para todos os caracteres conhecidos de línguas vivas.

Porque são usadas 3 versões? Resposta simples e direta: tamanho dos arquivos de fontes.

Para contornar este problema, é possível fazer duas coisas:

a) Indicar no cabeçalho qual é o tipo de codificação que está sendo utilizada;
b) Usar tags de acentuação XHTML

Ambos os métodos serão apresentados e, em geral, deve-se usá-los sempre.

4.1. Indicação de Codificação de Página

A primeira forma de resolver o problema é deixar claro para o navegador qual é o tipo de codificação sendo usada. A maneira de fazer isso é através de uma tag de cabeçalho chamada **<meta />**, que serve para definir algumas características do próprio documento.

Assim como já foi visto para a tag <html>, a tag <meta> também admite modificadores: *http-equiv* para indicar o nome da propriedade sendo definida e *content* para indicar o valor desta propriedade. Foge ao escopo do curso detalhar todas as possíveis configurações para a tag <meta />, mas esta em específico pode ser definida da seguinte forma:

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
```

Esta tag diz que a propriedade "Content-Type" (Tipo de Conteúdo) deve ser definida como "text/html; charset=iso-8859-1", isto é, um "*texto no formato html* seguindo a codificação de nome *iso-8859-1*". Esta codificação ISO-8859-1 é a codificação padrão que o Windows usa para computadores em língua portuguesa do Brasil.

Mas e se quiséssemos indicar que a codificação de nossa página é em UTF-8? Simples! A tag seria muito parecida:

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

Diferentemente da maioria das tags vistas anteriormente, a tag <meta /> também é pontual, isto é, deve ser sempre definida com uma / antes do >.

4.2. Tags de Acentuação XHTML

Apesar da estratégia anterior ser a mais amplamente usada, não é a única e, em tese, não deveria ser a única a ser utilizada pelos desenvolvedores. Há duas razões para isso:

a) O computador do cliente pode não entender a codificação indicada
b) O computador onde a página é criada pode não usar a codificação indicada

Nestes dois casos, há um problema. No primeiro deles o usuário não verá o texto corretamente, embora você pense que está tudo bem (em seu navegador tudo aparecerá perfeitamente). No segundo, nem mesmo você verá o texto adequadamente.

Para solucionar este dilema, o **XHTML tem uma maneira de indicar caracteres de acentuação**, de forma que os navegadores do mundo todo possam apresentar os símbolos de acentuação corretamente, quase que independentemente de outros fatores.

> Caso o cliente não possua instalada nenhuma fonte com os caracteres adequados, eles aparecerão incorretamente. Entretanto, isso tem se tornado relativamente incomum, dado que a maioria dos navegadores já vem com pelo menos uma fonte Unicode, incluindo a grande maioria dos caracteres usados no mundo todo.

A maneira de indicar estes caracteres é através de um código especial, que começa com o caractere &, é seguido pela letra que recebe o acento e, finalmente, temos o nome do acento - em francês, terminando a seqüência com um caractere ;. Por exemplo:

Tipo de Acento	Exemplo	Resultado
- Agudo: &_acute;	é	é
- Crase: &_grave;	À	À
- Circunflexo: &_circ;	ô	ô
- Trema: &_uml;	ü	ü
- Tilde: &_tilde;	ã	ã
- Cedilha: &_cedil;	ç	ç

5. IMAGENS NA PÁGINA WEB

Conceitos Chave:
- Grande atrativo das páginas web x Tempo de carregamento
-
    * title:

Um dos grandes atrativos da Web sempre foi sua capacidade de apresentar imagens. O uso de imagens torna uma página mais interessante, além de permitir a explicação de um assunto de forma mais elucidativa. Por outro lado, o uso exagerado e inadequado de imagens pode tornar o carregamento de uma página excessivamente lento. Assim, é importante saber usar as imagens e usá-las com parcimônia.

A tag usada para inserir uma imagem é a tag **<img />**. A tag <img /> (de IMaGe, ou IMaGem em português) é pontual, o que significa que ela precisa terminar com **/>**. Entretanto, a tag de imagem exige pelo menos um parâmetro, que indica **onde** está essa imagem, que pode estar tanto no disco, juntamente com o arquivo XHTML da página ou até mesmo em outro lugar da web. O parâmetro usado para indicar a localização da imagem é o parâmetro **src** (de SouRCe, que significa "origem", em inglês). Assim, o uso mais comum de uma figura é feito da seguinte forma:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                <img src="aflogo.gif" />
        </body>
</html>
```

O que será apresentado da seguinte forma:

Um outro parâmetro importante (e obrigatório no modo **strict**) é o parâmetro **alt**, que indica um texto curto a ser apresentado no lugar da figura, caso o navegador esteja sendo usado em um dispositivo em que não deve mostrar figuras (seja porque o aparelho não

suporta figuras, seja por razões de segurança). O texto do *alt* é usado em navegadores para deficientes visuais, pois é o texto do *alt* que é lido para descrever a figura. Assim, para ser correta, a página deve ser definida como indicado abaixo:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                <img src="aflogo.gif" alt="Logotipo Amusement Factory" />
        </body>
</html>
```

Se for executada num navegador modo texto, será apresentada da seguinte forma:

```
[IMG] Logotipo Amusement Factory
```

Alguns navegadores permitem que as imagens sejam "desligadas" para permitir uma navegação mais rápida. Também nestes casos o texto ALT será utilizado e a imagem só será carregada se o usuário clicar no texto da imagem.

Em alguns casos, queremos definir um texto explicativo mais longo para uma figura, caso o usuário deixe o ponteiro do mouse sobre a figura por alguns instantes, quando então a explicação aparecerá em um pequeno "balão de ajuda". Isso é útil, por exemplo, quando queremos descrever quem são as pessoas em uma foto, mas o número de pessoas é muito grande e não queremos gastar "espaço" da página com isso. Para atingir este objetivo, podemos usar o parâmetro **title**, conforme indicado a seguir:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                <img src="aflogo.gif" alt="Logotipo Amusement Factory"
                        title="Logotipo da Empresa do Professor" />
        </body>
</html>
```

5.1. Acelerando o Carregamento de Páginas com Imagens

Conceitos Chave:
- Dependência do tamanho da imagem para definição do layout
- Especificação do tamanho da imagem no XHTML
  * width:
  * heigth:
- Possíveis problemas: distorções, tempos de carregamento altos etc.

Muita vezes, quando o navegador vai carregar uma página com muitas imagens, ele não é capaz de mostrar **nada** da página até que carregue todas as imagens. isso ocorre porque, para distribuir e desenhar o conteúdo textual da página, o navegador precisa primeiro saber qual será o tamanho e posição das figuras que serão apresentadas.

Entretanto, para saber o tamanho das figuras, ele precisa carregá-las primeiro, o que pode ser bem demorado, dependendo do número e do tamanho das imagens. Porém, existe uma forma de ajudar o navegador a saber o tamanho das imagens, fazendo com que ele apresente o texto da página tão logo termine de carregar o arquivo .html, antes mesmo de baixar as imagens contidas na página.

Para conseguir isso, no HTML antigo, bastava indicar a largura da imagem pelo parâmetro width e a altura da imagem, pelo parâmetro height. Esta utilização é apresentada no exemplo a seguir (em HTML 4.1):

```
<HTML LANG="pt-BR">
    <HEAD>
        <TITLE>Teste de Imagem 2</TITLE>
    </HEAD>
    <BODY>
        <P><IMG src="aflogo.gif" width="330" height="80"
        TITLE="Logotipo da Empresa do Professor" ALT="Logotipo Amusement Factory">
        Logotipo do site do professor.</P>
    </BODY>
</HTML>
```

Caso a página seja carregada em uma conexão lenta, será possível observar duas etapas. Na primeira delas, o seguinte conteúdo será apresentado:

Logotipo do site do professor.

Sendo, então, a imagem preenchida posteriormente:

Logotipo do site do professor.

É claro que, em uma página com uma única (e pequena) figura, a diferença de tempo entre a apresentação do texto é praticamente imperceptível. Entretanto, em páginas com muitas figuras grandes, a diferença é bastante sensível e recomenda-se que sempre sejam indicadas as dimensões da figura.

Esse recurso, entretanto, **deve ser evitado em XHTML strict**, embora não seja considerado incorreto. Os parâmetros **width** e **height** podem ser usados com a tag img em XHTML strict, mas é mais indicado acrescentar um parâmetro **id** à imagem e usar o **id** no CSS para indicar a largura e altura da imagem (será visto posteriormente).

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                <img id="aflogo" src="aflogo.gif" alt="Logotipo Amusement Factory"
                        title="Logotipo da Empresa do Professor" />
        </body>
</html>
```

De qualquer forma, o que acontece se as dimensões forem indicadas de maneira "incorreta" no CSS? Neste caso, a imagem será redimensionada pelo navegador para aparecer do tamanho especificado. Observe o que ocorre quando indicamos uma largura duas vezes maior que o tamanho original:

Logotipo do site do professor.

Entretanto, convém lembrar que não é adequado redimensionar imagens desta forma, a não ser em casos muito especiais. Esse comentário é relevante no caso de redução de imagens. Quando ela é reduzida pelo CSS, ainda assim a imagem grande é transferida pela

rede, gastando muita banda da rede. Se a imagem irá aparecer pequena, o ideal é que ela seja reduzida por meio de um editor gráfico, como o Photoshop ou Corel Draw, de maneira que ela seja transmitida já em tamanho reduzido.

Para aumentar uma imagem o recurso pode ser interessante, mas tem limitações. O redimensionamento feito pelo navegador nem sempre tem um aspecto agradável, como poderia ser conseguido com um editor gráfico. Sendo assim, é importante analisar bem a situação antes de usar o redimensionamento das imagens pelo CSS. Em geral, se usarmos esse recurso, indicaremos o tamanho exato da imagem no CSS.

6. OS LINKS WEB

Conceitos Chave:
- Endereço Web: **www.google.com.br** (incompleto!)
  * *Home Page*: www.google.com.br/index.html
- Link: marcação que permite que um texto aponte para outra página
  * http://nome_de_computador/diretorio/arquivo.html
- Exemplo: http://www.terra.com.br/portal/index.html

Quando alguém decide que vai viajar em suas férias, uma das primeiras coisas que esta pessoa precisa observar são os seus *possíveis destinos*, não é?

Na Web trabalhamos com a metáfora da *navegação*, isto é, dizemos que nossos usuários *navegam* na Web. Se considerarmos que cada página na Web é um ponto de parada de navegação, isto é, um *ancoradouro*, precisamos indicar para quais outras páginas o usuário poderá navegar, a partir dali.

Assim, em qualquer página da Web encontraremos indicações de algumas páginas que poderão ser visitadas em seguida. Estas indicações são chamadas *ligações* ou *links*. Assim, um *link* é um elo entre a página atual e uma outra página qualquer que esteja disponível na WWW.

Este *link* pode ser apresentado como um texto ou uma imagem, mas sempre tem que indicar um *endereço na internet* (também conhecido como "ancoradouro principal"). São exemplos de endereço:

www.uol.com.br
www.terra.com.br
www.google.com.br
www.yahoo.com.br
www.hotmail.com
www.w3.org

Estes endereços simplificados indicam apenas o nome de um computador na rede, e não o nome da página. Quando um endereço é fornecido neste formato, um programa que roda no servidor da página, o *servidor web*, carrega uma página padrão, chamada *home page*.

Cada servidor web pode definir uma página diferente como *home page*, mas o comportamento mais comum é o seguinte: sempre que for indicado apenas o nome de um computador ou o nome de um computador e um diretório, o servidor web irá carregar a página **index.html**. Assim, no caso do Google, por exemplo, é possível o endereço www.google.com.br, e isso fará com que nosso navegador aporte no seguinte ancoradouro:

www.google.com.br/index.html

E este sim é o nome completo do *endereço web*, ou seja, o *endereço da página*. A forma completa de especificar um endereço principal é:

nome_de_computador/diretorio/arquivo.html

Observe como ele indica o caminho a ser seguido até chegar ao arquivo de uma página web. No caso do Terra, o endereço completo da home page é:

www.terra.com.br/portal/index.html

Neste caso, o endereço indica "abra o arquivo **index.html** que se encontra no diretório **portal** do servidor **www.terra.com.br**".

Serão estes *endereços* que usaremos para indicar os possíveis destinos a partir de uma página web.

6.1. Definindo Links em XHTML

Conceitos Chave:
- Marcação de Link: <a>...
    * href: Texto do Link
    * Vai para o UOL

Ok, então cada arquivo HTML na web tem um endereço específico e precisamos usá-lo para "*linkar*" uma página. Um link será um marcador, definido pela tag **<a>** ... **** (de *Anchor*, ou âncora), que faz uma **referência a um documento XHTML** (Html REFerence, ou HREF). A tag <a> tem início e fim, pois o texto (ou imagem) que estiver marcado por ela se tornará um *link*!

Nota: o nomê da tag <a>, Âncora, vem da já revelada analogia com a navegação.

Assim, todo link para uma outra página tem pelo menos 3 componentes: a indicação da tag **a**, o endereço de uma outra página indicada pelo modificador **href** e o texto (ou imagem) que representará o link, como indicado no trecho de código-exemplo a seguir:

```
<a href="http://www.uol.com.br/">Um Link</a>
```

O que será apresentado pelo navegador assim:

Um Link

O código completo segue:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                <p>
                        <a href="http://www.uol.com.br/">Vai para o UOL</a>:
                        UOL, um dos maiores portais do Brasil na Internet.
                </p>
        </body>
</html>
```

Este código será apresentado da seguinte forma:

Vai para o UOL: UOL, um dos maiores portais do Brasil na Internet.

E, clicando no link, o navegador será redirecionado corretamente para o site do UOL! Bastou indicar corretamente o endereço do ancoradouro destino e pronto!

Convém aqui fazer um comentário acerca do texto que é convertido em link. Muitas vezes vemos na Internet links como "Clique Aqui" ou "Download". Todos que programam páginas web fazem isso, mas é importante lembrar que esse tipo de coisa deve ser evitada.

A primeira razão para isso é filosófica: na filosofia do hipertexto, você não deve ter que modificar um texto para inserir links. Em outras palavras, o texto deve ser escrito como se não existisse link algum, e os links deviam ser naturalmente associados à palavras já existentes.

A segunda razão para isso é social: navegadores para deficientes visuais e deficientes físicos, que dependem da voz para selecionar links, normalmente apresentam uma lista de links em separado (seja na forma textual, seja na forma verbal), e o usuário dita qual dos links quer entrar. Agora, imagine se a lista de links tiver essa cara:

Clique Aqui
Clique Aqui
Clique Aqui
Download
Clique Aqui
Clique Aqui
Download
Clique Aqui
Download

Fica um tanto complicado de selecionar, não é? Por estas razões, tentemos sempre usar textos elucidativos nos links. Acredite em uma coisa: se o usuário está habituado à internet (e hoje todos estão), se existe um trecho diferenciado no meio de um texto, ele já sabe que é um link e sabe que pode clicar lá. Ninguém precisa dizer para ele "clique aqui".

6.2. Links com Títulos Explicativos

Conceitos Chave:
- title: ...
    * Tamanho de arquivo, tempo de download, detalhamento do conteúdo etc.

Assim como foi visto no caso das figuras, é possível adicionar um texto explicativo ao link, de forma que este texto só seja apresentado, dentro de um pequeno "balão", se o usuário mantiver o ponteiro do mouse sobre o link por alguns instantes.

Este tipo de texto é interessante para que o usuário obtenha mais informações sobre o que vai encontrar "do outro lado do link", ou seja, no próximo ancoradouro, para saber se vale a pena navegar até lá. Um uso comum e útil deste recurso é indicar o formato de um arquivo de mídia, seu tamanho em megabytes, tempo de download, dentre outros.

Para conseguir isso, usa-se o parâmetro modificador **title** dentro da tag <a>.... Ele é usado da seguinte forma:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
                <p>
                        <a href="http://www.uol.com.br/" title="Universo OnLine">
                        Vai para o UOL</A>:  UOL, um dos maiores portais do Brasil.
                </p>
        </body>
</html>
```

O que será apresentado da seguinte forma:

Vai para o UOL: UOL, um dos maiores portais do Brasil.

Observe que a aparência não muda em nada. Entretanto, se repousarmos o ponteiro do mouse por alguns segundos sobre este link, uma balãozinho de informações aparecerá com o texto "Universo OnLine", que definimos como o título do link.

7. LINKS INTERMEDIÁRIOS

Conceitos Chave:
- Ponto de Ancoragem: local onde o navegador pode parar dentro de uma página
- Todo ponto de ancoragem tem um nome.
    * Ex.: http://www.endereco.com/pagina.html#nome_da_posicao
    * Ex.: http://pt.wikipedia.org/wiki/Engenharia_de_software#Processo_de_Software

Como você deve ter observado, todos os links apresentados aqui levam ao topo de uma página. Por outro lado, você já deve ter observado que alguns sites usam links que apontam diretamente para conteúdo **no meio** de uma página. Como isso é possível?

O processo é exatamente o mesmo de criar um link tradicional, mas é preciso adicionar uma informação a mais no endereço da página, uma informação que indique para o navegador até onde ele deve rolar a página.

Isso é feito com o indicador "jogo da velha", isto é, o símbolo #. Este símbolo deve ser seguido do **nome** do ponto no qual o navegador deve parar a rolagem, chamados "***pontos de ancoragem***". Abaixo segue um exemplo genérico:

www.endereco.com/pagina.html**#nome_da_posicao**

Mais adiante veremos como indicar um *ponto de ancoragem* em uma página, mas no momento vejamos como usar o indicador #. Vamos ver isso com base em um exemplo. A página de Engenharia de Software na WikiPedia (http://pt.wikipedia.org/wiki/Engenharia_de_software) tem um ponto de rolagem chamado **Processo_de_Software**.

Assim, para indicar o link **direto** para o ancoradoro de nome "Processo_de_Software", basta indicar o endereço como se segue:

http://pt.wikipedia.org/wiki/Engenharia_de_software#Processo_de_Software

No documento HTML, isso fica da seguinte forma:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
        <p>
        <a href="pt.wikipedia.org/wiki/Engenharia_de_software#Processo_de_Software">
        Processo de Software</a>: Direto para a WikiPedia.
        </p>
        </body>
</html>
```

O que será apresentado da seguinte forma:

Processo de Software: Direto para a WikiPedia.

7.1. Definindo um Ancoradouro no Meio de uma Página Web

Conceitos Chave:
- id:
    * Deve ser colocado exatamente na posição que ficará no topo da página

Bem, parece simples adicionar o link para o meio de uma página, mas e se quisermos definir um ancoradouro no meio de nossa própria página?

Isso também é bastante simples e, para isso, também se usa a tag **<a>**, que marcará o ponto em que o navegador irá parar de rolar a página**.** Neste caso, entretanto, não será usado o parâmetro href e sim o parâmetro **id**, que definirá o nome deste ponto, ou seja, o nome que deverá ser indicado no endereço Web para que o navegador role até este ponto da página.

Neste caso, mesmo que não haja texto, **é obrigatório** "fechar" a *tag* com o ****. Assim, para definir um ancoradouro intermediário chamado "Meu_Perfil" no meio de uma página, use a seguinte construção XHTML:

pagina.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EM"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pt-BR>
        <head>
        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-18" />
        <title>Título da página Web!</title>
        </head>
        <body>
        <p>     Aqui come&ccedil;a a p&aacute;gina, com uma s&eacute;rie de
                informa&ccedil;&otilde;es interessantes que se estendem por uma
                centena de linhas e tal.      </p>
        <a id="Meu_Perfil"></a>
        <p>     Aqui come&ccedil;a o seu perfil.    </p>
        <p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p>
        <p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p>
        <p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p>
        <p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p><p>.</p>
        </body>
</html>
```

O que será apresentado da seguinte forma, se for digitado seu endereço, como por exemplo http://www.aluno.com/index.html :

Aqui começa a página, com uma série de informações interessantes que se estendem por uma centena de linhas e tal.

Aqui começa o seu perfil.

Note que nada apareceu no lugar em que definimos o ancoradouro. Entretanto, se o usuário entrar na sua página com o endereço http://www.aluno.com/index.html#Meu_Perfil, o resultado será o apresentado abaixo:

Aqui começa o seu perfil.

O usuário poderá rolar a tela para cima e ainda verá o texto introdutório, mas ao entrar, ele caiu diretamente em seu perfil.

8. LISTAS EM XHTML

Conceitos Chave:
- Listas Ordenadas
- Não-Ordenadas
- Listas de Definições

Um recurso bastante usado do XHTML é a capacidade de exibir listas. Existem basicamente dois tipos de listas: as ordenadas (numeradas) e as não ordenadas (listas de "bullets"). Por exemplo:

Lista Ordenada:

1. Primeiro nível
1.1. Segundo nível.
2. Primeiro nível novamente.
...

Lista Não Ordenadas:

- Primeiro item
- Segundo item
- Terceiro item
...

Existem ainda as Listas de Definição:

Microprocessador
    Circuito usado em computadores para processar dados.
Sinal
    Pulso de tensão elétrica específico para acionar um dispositivo.
...

8.1. Listas Não Ordenadas

Conceitos Chave:
- Tag demarcador de Lista Não-Ordenada
    * ...
- Tag demarcador de Item de Lista
    * ...
- Exemplo

Uma lista não ordenada sempre será demarcada pelas tags **
** ... **
** (UL = Unordered List, ou Lista Não Ordenada). Adicionalmente, cada elemento de lista deve ser delimitado pelas tags **
** ... **
** (LI = List Item, ou Item de Lista).

Assim, se quisermos indicar uma lista não ordenada, em XHTML a especificaremos da seguinte maneira:

```
<ul>
        <li>Um item.</li>
        <li>Outro item.</li>
        <li>Mais outro item.</li>
</ul>
```

O que será apresentado assim:

- Um item.
- Outro item.
- Mais outro item.

É possível indicar uma lista dentro de outra:

```
<ul>
        <li>Um item.</li>
        <ul>
                <li>Um sub-item.</li>
                <li>Outro sub-item.</li>
        </ul>
        <li>Mais outro item.</li>
</ul>
```

O que será apresentado assim:

- Um item.
  - Um sub-item.
  - Outro sub-item.
- Mais outro item.

8.2. Listas Ordenadas

Conceitos Chave:
- Tag demarcador de Lista Ordenada
  * ...
- Tag demarcador de Item de Lista
  * ...

As listas ordenadas são exatamente iguais às listas não-ordenadas, mas ao invés de serem demarcadas pelas tags <ul> ... , são demarcadas pelas tags **
** ... **
** (OL = Ordered List, ou Lista Ordenada). Os elementos de lista também devem ser delimitados pelas tags **
** ... **
**. Assim, se quisermos indicar uma lista ordenada, em XHTML a especificaremos assim:

```
<ol>
        <li>Um item.</li>
        <li>Outro item.</li>
        <li>Mais outro item.</li>
</ol>
```

O que será apresentado assim:

1. Um item.
2. Outro item.
3. Mais outro item.

É possível indicar uma lista dentro de outra:

```
<ol>
        <li>Um item.</li>
        <ol>
                <li>Um sub-item.</li>
        </ol>
        <li>Mais outro item.</li>
        <ul>
                <li>Outro sub-item.</li>
        </ul>
</ol>
```

O que será apresentado assim:

1. Um item.
   1. Um sub-item.
2. Mais outro item.
   - Outro sub-item.

8.3 Listas de Definição

Conceitos Chave:
- Tag demarcador de Lista de Definição
  * <dl> ... </dl>
- Tag demarcador de Termos
  * <dt> ... </dt>
- Tag demarcador de Descrição
  * <dd> ... </dd>
- Exemplo

As listas de definição são usadas, por exemplo, para fazer glossários. Sua função é apresentar termos e sua explicação. A lista deve ser demarcada pelas tags **<dl>**...**</dl>** (de Definition List). Os termos são demarcados pelas tags **<dt>**...**</dt>** e a descrições por **<dd>**...**</dd>**.

```
<dl>
        <dt>Microprocessador</dt>
        <dd>Circuito usado em computadores para processar dados.</dd>
        <dt>Sinal</dt>
        <dd>Pulso de tensão elétrica específico para acionar um dispositivo.</dd>
</dl>
```

O que será apresentado da seguinte forma:

Microprocessador
        Circuito usado em computadores para processar dados.
Sinal
        Pulso de tensão elétrica específico para acionar um dispositivo.

9. TAGS DIVERSAS DE MARCAÇÃO

Além das tags já apresentadas, existe ainda um importante conjunto de tags a serem apresentados. Serão abordadas, a seguir, a maior parte delas, incluindo a importante tag de tabelas. Entretanto, um conjunto muito importante de tags, as de formulário, serão deixadas para o futuro.

<abbr>...</abbr> - Usado para indicar uma abreviatura, como por exemplo, <abbr>Prof.</abbr>.

<acronym>...</acronym> - Usado para indicar uma sigla, como por exemplo, <acronym>GNU</acronym>.

<address>...</address> - Usado para marcar o endereço (de e-mail, por exemplo) do autor da página. Por exemplo: <address>Rua do Limoeiro, 37</address>.

<base>...</base> - Muda a referência dos links de uma página. Pode ser usado com modificador HREF ou TARGET.

<bdo>...</bdo> - Especifica a direção do texto. O modificador *dir* pde ter os valores **rtl** ou **ltr**.

<big>...</big> - Usado para fazer com que um trecho do texto seja apresentado em letras maiores, ressaltadas.

<blockquote>...</blockquote> - Usado para marcar citações exatas longas.

<cite>...</cite> - Usado para marcar um texto como uma citação (média).

<code>...</code> - Usado para marcar um texto como sendo um código de programação.

<comment>...</comment> ou **<!- ... ->** - Usados para comentários que não devem ser exibidos pelo navegador.

~~...~~ - Usado para marcar um trecho do texto como não sendo mais válido (riscado).

<dfn>...</dfn> - Usado para marcar a definição de um termo.

** ... ** - Usado para marcar logicamente uma seção dentro de uma página HTML. Seu uso é muito importante e será visto nas aulas seguintes.

... - Usado para marcar um texto de forma que ele seja enfatizado.

<ins>...</ins> - Marca um texto que deve ser adicionado ao texto. Usado, normalmente, junto com os delimitadores ~~...~~.

<kbd>...</kbd> - Marca um texto que deve ser digitado pelo usuário, em alguma situação.

<link>...</link> - (tag de cabeçalho) Usado para indicar associação do documento atual com algum outro. Em geral usado para indicar folhas de estilo.

<meta>...</meta> - (tag de cabeçalho) Usado para indicar informações sobre a página web.

<noscript>...</noscript> - Usado para indicar um texto avisando ao usuário que o navegador dele precisa de suporte a script para que a página funcione.

<object>...</object> - Insere um elemento externo na página web, como um plugin, por exemplo.

<pre>...</pre> - Usado para marcar um texto pré-formatado. Dentro desta região, os "enters" do texto serão interpretados pelo navegador como quebras de linha.

<q>...</q> - Usado para citações exatas curtas.

<samp>...</samp> - Usado para marcar um texto como sendo um exemplo de código de programação.

<script>...</script> - (Tag de Cabeçalho) Serve para indicar um script para ser usado na página.

<small>...</small> - Usado para fazer com que um trecho do texto seja apresentado em letras menores.

... - Usado para marcar que trecho de um texto deve estar bastante ressaltado.

<style>...</style> - Usado para indicar um estilo de formatação dentro da própria página html. Evitar. É melhor usar folhas de estilo externas.

_{...} - Usado para colocar índices inferiores (subscrito).

^{...} - Usado para colocar índices superiores (sobrescrito).

<var>...</var> - Usado para marcar uma palavra como uma variável de programa ou uma parte variável de um texto.

10. TAGS DEPRECIADAS ("PROIBIDAS")

... - Texto em negrito. Não é uma tag proibida, mas deve ser evitada. Prefira .

... Texto em itálico. Não é depreciada, mas deve ser evitada. Prefira .

<applet>...</applet> - Insere um elemento externo na página web, como um plugin, por exemplo. Tag depreciada. Use <object>.

<basefont> - Modifica a fonte padrão da página. Tag depreciada. Use CSS.

** **...** ** - Centraliza o texto. Tag depreciada. Use CSS.

<dir>...**</dir>** - Marca uma lista de diretório. Tag depreciada. Use <ul>.

****...**** - Muda a fonte usada em um texto. Tag depreciada. Use CSS.

<iframe>...</iframe> - Carrega uma outra página em uma área da sua página. Não é exatamente padrão, embora seja suportado pela maioria dos navegadores. Tag depreciada. Use CSS.

<menu>...**</menu>** - Marca uma lista de menu. Tag depreciada. Use <ul>.

<nobr>...</nobr> - Usado para indicar um trecho de texto que o navegador não deve quebrar no fim de linha. Tag depreciada. Use CSS.

<s>...</s> Corta um texto. Esta tag é depreciada. Use <del>.

<strike>...</strike> Corta um texto. Esta tag é depreciada. Use <del>.

<u>...</u> Texto sublinhado. Esta tag é depreciada. Use ou , de acordo com a situação.

<wbr>...</wbr> - Usado para indicar locais possíveis de quebra de palavra. Usado normalmente dentro de um <nobr>...</nobr>. Tag depreciada. Use CSS.

11. TABELAS

Um recurso muito útil e importante no XHTML - porém freqüentemente muito mal utilizado - é o de apresentação de tabelas. Para que uma tabela seja apresentada adequadamente e rapidamente pelo navegador, ela precisa estar completamente definida, algo que muitos programadores XHTML se esquecem de fazer.

Uma tabela é basicamente demarcada pelas tags **<table>** ... **</table>**. Dentro destas tags, temos três seções: a seção **<thead>...</thead>**, onde devem ser colocados os cabeçalhos da tabelas, a seção **<tbody>** ... **</tbody>**, onde ficam as linhas de informação da tabela e também a seção **<tfoot>...</tfoot>,** onde devem ficar o rodapé da tabela. Estes marcadores de seções são opcionais, mas são altamente recomendados para facilitar a aplicação de estilos no futuro. É comum que se remova, entretanto, a região <tfoot>, por não ter função dentro de uma tabela específica.

Logo em seguida à tag <table> e antes de qualquer outra, deve ser indicada a tag **<caption>** ... **</caption>**, que serve para indicar a legenda da tabela. Com estes elementos posicionados, o código fica como especificado a seguir:

```
<table>
        <caption>Tabela 1: Uma tabela</caption>
        <thead>
        ...
        </thead>
        <tbody>
        ...
        </tbody>
        <tfoot>
        ...
        </tfoot>
</table>
```

Dentro de cada uma das seções <thead>, <tbody> ou <tfoot> cada linha da tabela tem sua estrutura própria e deve ficar demarcada pelas tags **<tr>** ... **</tr>** (Table Row, ou Linha de Tabela). Assim, se nossa tabela terá 3 linhas, podemos escrever sua estrutura da seguinte forma:

```
<table>
        <caption>Tabela 1: Uma tabela</caption>
        <thead>
        <tr>
        ... [ linha 0 ]
        </tr>
        </thead>
```

```
        <tbody>
        <tr>
        ... [ linha 1 ]
        </tr>
        <tr>
        ... [ linha 2 ]
        </tr>
        <tr>
        ... [ linha 3 ]
        </tr>
        </tbody>
</table>
```

Dentro das linhas iremos colocar as "células" de nossa tabela. Uma célula pode ser de um de dois tipos: uma **célula título** ou uma **célula de dados**. No primeiro caso, delimitamos a informação com as tags **<th>** ... **</th>** (de Table Heading). No segundo, em caso de células de dados, delimitamos a informação com as tags **<td>** ... **</td>** (de Table Data).

Assim, se na primeira linha tivermos títulos de coluna e nas outras duas linhas tivermos dados, numa tabela com 2 colunas, o código fica:

```
<table>
        <caption>Tabela 1: Uma tabela</caption>
        <thead>
        <tr>
                <th>Título Coluna 1</th>
                <th>Título Coluna 2</th>
        </tr>
        </thead>
        <tbody>
        <tr>
                <td>Coluna 1, Linha 1</td>
                <td>Coluna 2, Linha 1</td>
        </tr>
        <tr>
                <td>Coluna 1, Linha 2</td>
                <td>Coluna 2, Linha 2</td>
        </tr>
        </tbody>
</table>
```

O que será apresentado da seguinte forma (lembrando que, por padrão, as linhas das tabelas não vão aparecer. Veremos como acrescentar as linhas posteriormente):

Tabela 1: Uma tabela

Título Coluna 1	Título Coluna 2
Coluna 1, Linha 1	Coluna 2, Linha 1
Coluna 1, Linha 2	Coluna 2, Linha 2

Lembrando aqui que é possível colocar uma tabela dentro de outra, como é apresentado no código a seguir.

```
<table>
	<caption>Tabela 2: Uma tabela com outra dentro</caption>
	<thead>
	<tr>
		<th>Título Coluna 1</th>
		<th>Título Coluna 2</th>
	</tr>
	</thead>
	<tbody>
	<tr>
		<td>
		<table>
			<tbody>
				<tr>
					<th>Título Sub Coluna 1</th>
					<th>Título Sub Coluna 2</th>
				</tr>
				<tr>
					<td>Sub Coluna 1, Linha 1</td>
					<td>Sub Coluna 2, Linha 1</td>
				</tr>
			</tbody>
		</table>
		</td>
		<td>Coluna 2, Linha 1</td>
	</tr>
	<tr>
		<td>Coluna 1, Linha 2</td>
		<td>Coluna 2, Linha 2</td>
	</tr>
	</tbody>
</table>
```

E o resultado será como o apresentado abaixo:

Tabela 2: Uma tabela com outra dentro

Título Coluna 1	**Título Coluna 2**	
Coluna 1, Linha 1	**Título Sub Coluna 1**	**Título Sub Coluna 2**
	Sub Coluna 1, Linha 1	Sub Coluna 2, Linha 1
Coluna 1, Linha 2	Coluna 2, Linha 2	

Outra possibilidade é expandir uma linha por duas colunas, usando o modificador **colspan** dentro da tag TD ou TH. ou seja: para obter a aparência a seguir, use colspan como aparece no código em seguida.

Tabela 3: Uma tabela com coluna expandida

Título das Colunas	
Coluna 1, Linha 1	Coluna 2, Linha 1

Observe, no código a seguir, o uso de colspan dentro da tag <th>. O número 2 indica o número de colunas que aquela célula deve ocupar:

```
<table>
	<caption>Tabela 3: Uma tabela com coluna expandida</caption>
	<thead>
	<tr>
		<th colspan="2">Título das Colunas</th>
	</tr>
	</thead>
	<tbody>
	<tr>
		<td>Coluna 1, Linha 1</td>
		<td>Coluna 2, Linha 1</td>
	</tr>
	</tbody>
</table>
```

O mesmo vale para estender uma célula para ocupar mais de uma linha, bastando usar o modificador **rowspan**.

12. FORMULÁRIOS: O QUE SÃO?

Conceitos Chave:
- Tag <form>...</form>
- Parâmetros
  * action = "pagina_destino"
  * method = "post"
  * id = "form1"
- Campos possuem nome!

Formulários são conjuntos de campos de entrada de dados que permitem que o conteúdo destes campos sejam enviados para um endereço específico, onde eles serão processados.

Um exemplo de um formulário:

A tag que permite a criação de formulários é a tag <form>.

```
<form>
    [... conteúdo do formulário ...]
</form>
```

A tag de formulário precisa indicar o que precisa ser feito quando o usuário clicar no botão "enviar". Isso pode ser feito com o parâmetro "action" da tag **form**. Neste parâmetro podemos indicar uma função de javascript (veremos isso no futuro) ou uma outra página, que terá a função de processar os dados enviados (veremos isso no futuro, também).

Assim, o primeiro parâmetro pode ser especificado como indicado a seguir.

```
<form action="pagina.php">
    [... conteúdo do formulário ...]
</form>
```

Entretanto, como foi dito anteriormente, há duas formas de enviar dados: pelo método GET ("linha de comando") ou pelo método POST (usando um "arquivo de dados"). Bem, com os formulários podemos enviar por qualquer um dos modos, então é necessário indicar também o formato de envio na tag **form**, o que pode ser feito com o parâmetro method:

```
<form action="pagina.php" method="post">
    [... conteúdo do formulário ...]
</form>
```

O que diferencia o método POST do método GET? Basicamente, tudo que é enviado pelo método GET precisa ser colocado na URL. Como o tamanho de uma URL é relativamente limitado (varia de navegador para navegador, mas situa-se em torno de 4KB), se for necessário enviar muitos dados ou um arquivo, o método GET se torna inadequado. Adicionalmente, o método GET é menos seguro, pois os dados poderão expostos na URL.

Por outro lado, o método GET é mais rápido, carrega menos o servidor e, como veremos, é muito útil no debugging de aplicações Web.

Agora, imagine que uma página tem vários formulários, todos com os mesmos campos. Por exemplo:

Formulário 1: Reclamação
- Campo 1: Nome
- Campo 2: E-Mail
- Campo 3: Dados

Formulário 2: Sugestão
- Campo 1: Nome
- Campo 2: E-Mail
- Campo 3: Dados

Quando o usuário aperta "enviar", talvez o programador queira processar o campo "Dados" do formulário 2... mas como fazer isso, se ambos os formulários possuem o mesmo campo? Para isso, é comum darmos **id**s (identificações) formulários. Por exemplo:

```
<form action="pagina.php" method="post" id="form1">
    [... conteúdo do formulário ...]
</form>
```

Desta maneira, se eu quiser me referir ao conteúdo do campo "Dados" deste primeiro formulário, eu posso indicá-lo como:

form1.Dados

Isso permite diferenciar campos com o mesmo nome que estejam em formulários distintos. No segundo formulário o mesmo campo seria acessado pelo seguinte nome:

form2.Dados

Note que, até o momento, falamos apenas da tag <form>...</form>, que delimita a área do formulário. Mas um formulário não é só isso! Um formulário precisa ter campos e botões de ação!

Minimamente, um formulário deverá conter um botão "Enviar", "Aplicar", "Atualizar"... ou seja, um botão que acionará a ação indicada no "action", que, como já dito, pode ser um método javascript que processará os dados ou mesmo um envio direto a um programa no servidor, com a indicação de um endereço.

Além deste botão, diversos outros elementos podem aparecer em um formulário, como campos te texto, caixas de seleção, listas de seleção etc. Cada um deles será apresentado a seguir.

12.1. Tags de Elementos de Formulário

Conceitos Chave:

- <input />
    * Type; Name; id; Value; AccessKey; Title; Dir...
    * Disabled
- Botões
    * <input type="submit" name="Texto" />
        + Reset; Button
- Entrada de Texto
    * <input type="text" value="valor_inicial" />
        + MaxLength; ReadOnly... password
- CheckBox
    * <input type="checkbox" value="valor" />
        + Checked
- RadioBox
    * <input type="radio" value="valor" name="nome" />
        + Checked

```
- ComboBoxes
        * <select name="nome" id="id">
                <option value="valor">Texto</option>
          </select>
                + Size; Multiple; Disable
                + Selected; Disable
- Área de Texto
        <textarea>...</textarea>
                + Rows; Cols; MaxLength; ReadOnly
- <fieldset> <legend>...</legend> ... </fieldset>
- <input type="hidden" name="nome" value="valor" />
- <input type="file" name="nome" />
        + <form action="pag.php" method="post" ENCTYPE="multpart/form-data">
```

Como dito anteriormente, um formulário pode conter diversos elementos internos. Cada um destes elementos é especificado por uma tag específica. Cada um deles será especificado a seguir, mas vale ressaltar desde já que a tag **<input />** é pontual, ou seja, deve ser grafada com fechamento **/>** e que será usada para a maioria dos controles de formulários.

Para especificar o tipo de controle, a tag **input** aceita o parâmetro **type**, que indica o tipo de elemento. Outros parâmetros comuns são o name, que indica o nome do elemento, e o parâmetro **value**, que indica o valor inicial de preenchimento.

Adicionalmente, há o parâmetro id, que serve para dar uma identificação única para um elemento (para uso com a tag **label**, por exemplo) e o parâmetro **accesskey**, que indica uma tecla de atalho para o elemento. Assim, o formato básico e genérico da tag **input** é:

```
<input type="tipo" name="nome" value="valor" id="id" accesskey="tecla" />
```

Todo elemento dentro de um formulário pode ser desligado, usando o parâmetro **disabled="disabled"**. A utilidade disso surge apenas quando associarmos os formulários ao uso de javascript.

Há outros parâmetros aceitáveis, como **title**, dentre outros, que possuem o mesmo uso visto anteriormente em outras tags. Os detalhes para cada tipo de controle serão especificados nas seções seguintes.

12.1.1. Botões

Como dito inicialmente, praticamente todos os formulários precisam ter um botão de envio, para que o usuário indique quando quer que as informações sejam transmitidas. Para incluir um botão deste tipo, é usada a tag **input**, já mencionada anteriormente. A sintaxe mais comum de um botão de envio é:

```
<input type="submit" value="texto_do_botao" />
```

Dificilmente se coloca um nome em um botão de envio, dado que só deve existir um por formulário, podendo ele ser acessado por *nome_do_formulario.submit* . Este botão, por padrão, executa a ação indicada pelo campo action da tag **form**.

Um outro tipo de botão comum é o botão "reset", que limpa todos os campos de um formulário, cuja sintaxe é descrita abaixo:

```
<input type="reset" value="texto_do_botao" />
```

Os botões podem ser usados para chamar métodos específicos de javascript, mas isso deve ser definido no código javascript. Para criar botões deste tipo, usa-se o tipo "button":

```
<input type="button" value="texto_do_botao" />
```

12.1.2. Campos de Texto

Os campos de texto são compostos por uma linha onde é possível digitar um texto qualquer, sendo que cada campo de texto deve ter seu próprio nome. A sintaxe é a seguinte:

```
<input type="text" value="valor_inicial" />
```

Isso faz aparecer na página o seguinte campo:

Uma versão alternativa é o tipo "password", que esconde os caracteres digitados:

```
<input type="password" value="valor_inicial" />
```

Note que tanto o tipo "text" quanto "password" possuem dois parâmetros especiais: o parâmetro **maxlength** e o parâmetro **size**. O parâmetro **maxlength** permite especificar o maior número de caracteres que um campo aceita. O parâmetro **size** especifica o comprimento da caixa de texto (parte visível da caixa de texto).

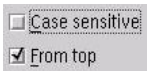
Finalmente, há o parâmetro **readonly**, que indica que o texto de um campo não pode ser modificado.

12.1.3. Caixa de Seleção (checkbox)

As caixa de seleção são controles que permitem você marcar algo como "sim" ou "não", ou seja, indicar opções que você deseja e desmarcar opções que você não deseja. A sintaxe básica é:

```
<input type="checkbox" value="valor" [checked="checked"] />
```

Onde o **checked="checked"** é um parâmetro adicional que indica se, inicialmente, a opção estará marcada ou não. O valor **value** será associado ao controle apenas se a caixinha estiver selecionada. Caso contrário, o valor associado ao controle será vazio. Exemplo de caixas de seleção:



12.1.4. Caixas de Opção (radiobox)

Os botões de opção são usados quando temos um pequeno número de alternativas fixas para uma mesma opção, e apenas uma delas pode ser selecionada de cada vez. Um exemplo de sintaxe seria:

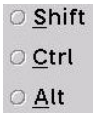
```
<input type="radio" name="sexo" id="fem" value="fem" [checked="checked"]  />
<input type="radio" name="sexo" id="masc" value="masc" />
```

Note que ambos os campos possuem o mesmo nome (name), já que ambos se referem à mesma seleção. O próprio navegador de encarrega de garantir que apenas um deles está selecionado de cada vez. Mais uma vez, a propriedade **checked** existe para indicar qual opção estará marcada inicialmente.

Um exemplo de uso de botões de opção segue abaixo:

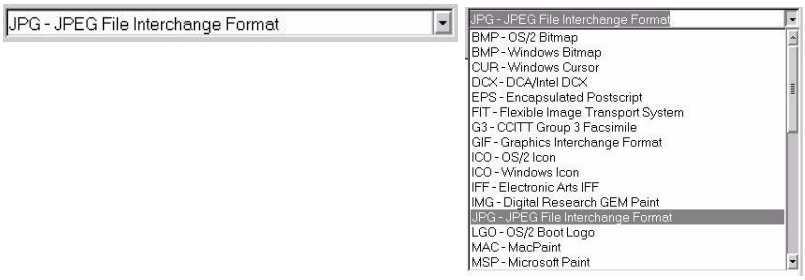
```
<input type="radio" name="key" id="shf" value="0" accesskey="S" />Shift<br />
<input type="radio" name="key" id="ctr" value="1" accesskey="C" />Ctrl<br />
<input type="radio" name="key" id="alt" value="2" accesskey="A" />Alt
```

Isso seria apresentado de maneira similar à indicada abaixo:



12.1.5. Lista de Seleção (combobox)

As listas de seleção são aquelas listas "drop-down", que apresentam apenas um valor mas, se clicarmos no botão à direita delas, uma lista maior é mostrada para que selecionemos um dos valores existentes. A seguir, temos um exemplo de uma lista de seleção:



Para criar uma lista deste tipo, a sintaxe é a seguinte:

```
<select name="nome_da_lista" id="id_da_lista>
        <option value="valor1">Valor 1: a</option>
                [...]
        <option value="valor2">Valor 2: b</option>
</select>
```

Cada linha <option>...</option> será uma opção da lista. O **value** de cada option é o valor que será associado à lista de seleção (valor selecionado) quando o conteúdo do formulário for enviado. O texto entre <option>...</option> é o texto que será apresentado na lista.

Tanto o campo de seleção <select>, quanto cada campo de opção <option> podem ser desligados com o parâmetro **disable="disable"**. Podemos ter uma caixa de seleção múltipla com o parâmetro **multiple="multiple"** na tag <select>. A tag <select> ainda aceita um parâmetro que identifica quantas linhas serão apresentadas ao clicar na lista de seleção: **size="numero_de_linhas"**. Para que uma opção inicie selecionada, basta usar o parâmetro **selected="selected"** dentro da tag <option> correspondente.

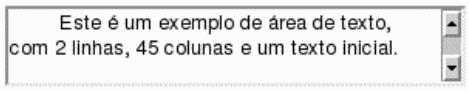
12.1.6. Área de Texto

As áreas de texto são usadas para que o usuário possa incluir um texto mais longo, da maneira que ele desejar. São usados, por exemplo, em formulários de envio de mensagens do tipo "fale conosco".

A indicação é simples, usando a tag <textarea>...</textarea>, sendo que o texto delimitado aparecerá, por padrão, na área de texto. Como o elemento de área de texto será apresentado na tela, é possível especificar seu tamanho, usando os parâmetros rows e cols para linhas e colunas respectivamente. Um exemplo pode ser verificado a seguir.

```
<textarea rows="2" cols="45">
        Este é um exemplo de área de texto, com 2 linhas, 45 colunas e um texto inicial.
</textarea>
```

E o resultado apresentado na tela será:



Este controle também aceita o parâmetro **maxlength**, que indica o número máximo de caracteres que é possível digitar na área de texto.

12.1.7. Conjunto de Controles

Em caixas de diálogo de aplicativos é muito comum que alguns controles sejam agrupados visualmente com o uso de uma borda que, eventualmente, possui um texto (legenda) no canto superior esquerdo. É possível criar este tipo de agrupamento em formulários XHTML, usando para isso a tag <fieldset>. A tag <legend> é usada para especificar o texto que aparece na borda a ser desenhada. O formato é:

```
<fieldset>
        <legend>Título da borda</legend>
                [...]
</fieldset>
```

Um exemplo mais completo segue abaixo:

```
<form>
        <fieldset>
                <legend>Dados Pessoais:</legend>
                        Name: <input type="text" SIZE="30" id="nome" /><br />
                        E-Mail: <input type="text" SIZE="30" id="email" /><br />
                        Data Nasc.: <input type="text" SIZE="10" id="nasc" /><br />
        </fieldset>
</form>
```

Que seria apresentado no documento da seguinte forma:

12.1.8. Dados "Escondidos"

Algumas vezes é preciso definir alguns valores em um formulário sem que os mesmos sejam apresentados para o usuário. Este valor pode indicar o tipo de formulário ou alguma instrução para o processamento posterior do formulário (indicando tratar-se de uma edição de dados já existentes, por exemplo).

Para este fim, existe o campo de entrada <input> do tipo **hidden**. Os parâmetros deste tipo de campo são, simplesmente, o nome do campo e o valor, respectivamente indicados pelos parâmetros name e **value**. É possível também indicar uma id para este campo. O formato padrão é indicado a seguir:

```
<input type="hidden" name="nome" value="valor" />
```

12.1.9. Campo de Envio de Arquivos

O campo de envio de arquivos deve ser especificado pela tag <input> do tipo **file**. Este controle será apresentado como um campo de texto e um botão "browse" ou "navegar", que serve para indicar um (ou vários) arquivo que será enviado junto com o restante do formulário. A sintaxe é simples:

```
<input type="file" name="arquivo" />
```

Isso, entretanto, não é suficiente para que o arquivo chegue "em paz" ao servidor. Como haverá dados binários sendo enviados com o formulário (ao invés de apenas texto), é preciso indicar na tag de declaração do formulário <form> que dados binários serão enviados também. Isso pode ser feito usando o parâmetro **enctype**, conforme indicado:

```
<form action="pagina.php" method="post" name="form1" enctype="multpart/form-data" />
    [... conteúdo do formulário ...]
</form>
```

12.2. Outras Tags para Formulários

Conceitos Chave:
- <label for="id">
- <button type="button">...</button>
- <optgroup label="Grupo">...</optgroup>
- <input type="image" src="icone.gif" width="32" height="32">

Além dos controles básicos já apresentados, existem algumas outras tags que podem ser usadas em formulários. Nesta seção veremos alguns destes tags.

12.2.1. LABEL

Em todos os exemplos apresentados, indicamos o texto de um campo como um "texto jogado". Existe uma forma mais correta de indicar os textos de campos de formuário, usando a tag **label**.

A tag <label> ... </label> marca um texto qualquer como sendo a "etiqueta" de um campo cujo id esteja definido. Isso é feito da seguinte forma:

```
<label for="id_do_elemento">Texto</label>
<input type="tipo" id="id_do_elemento">
```

Note que o valor do parâmetro **for** da tag **label** precisa ser **exatamente o mesmo** valor do parâmetro id ao qual ele se refere.

12.2.2. BUTTON

Além da tag **input** do tipo "button", que cria um botão simples, é possível usar a tag <button type="button"> em seu lugar. A tag **button**, diferentemente da tag **input**, permite coisas muito interessantes, pois ela define uma espécie de "sub página" dentro do botão. Isso significa que é como se o botão fosse um **div**, e é possível inserir qualquer tipo de código XHTML dentro deste botão. Exemplo:

```
<button type="button">
    <table>
        <tr><td>
            <img src="http://www.caetano.eng.br/main/images/aflogo_horiz.gif" alt="logo" />
        </td><td>
            <p>Um texto qualquer</p>
        </td></tr>
    </table>
</button>
```

O que será apresentado da seguinte forma:

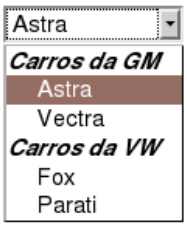
Observe a flexibilidade que esta tag proporciona, por permitir que praticamente se crie elementos XHTML dentro do botão e que, obviamente, podem ser configurados com o uso de CSS.

12.2.3. Grupos de Opção em Lista de Seleção (ComboBoxes)

Sempre que temos uma combobox com muitas opções, podemos organizá-las em grupos usando a tag <optgroup>...</optgroup>. Esta tag funciona da seguinte forma:

```
<select>
	<optgroup label="Carros da GM">
		<option value="astra">Astra</option>
		<option value="vectra">Vectra</option>
	</optgroup>
	<optgroup label="Carros da VW">
		<option value="fox">Fox</option>
		<option value="parati">Parati</option>
	</optgroup>
</select>
```

Isto será apresentado da seguinte forma:



12.2.4. Campo Imagem

O campo imagem serve para acrescentar uma figura ou ícone no formulário. Este campo é indicado com a tag **input**, com o tipo **image** identificado. Como uma imagem será apresentada, é preciso indicar o arquivo da imagem com o parâmetro **src**, além da largura e altura, usando os parâmetros **width** e **height** (que também podem ser definidos pelo CSS), conforme indicado a seguir:

```
<input type="image" src="icone.gif" width="32" height="32" />
```

Um outro uso para este tipo de campo é transformá-lo em um "botão-imagem". Para isso é preciso dar também um nome para o campo, usando o parâmetro name, para que se possa associar uma função ao evento de "clicar na imagem" usando JavaScript.

```
<input type="image" src="botao.gif" name="ok" width="32" height="32" />
```

13. BIBLIOGRAFIA

W3C: http://www.w3.org/

RAMALHO, J.A. *HTML 4 Prático e Rápido*. Editora Berkeley, 1999.

BOENTE, A. *Programação Web Sem Mistérios*. Editora Brasport, 2006.

NIELSEN, J. *Projetando Websites*. Ed. Campus, 2000.