GOVERNO DO ESTADO DO CEARÁ

Secretaria da Educação

ESCOLA ESTADUAL DE EDUCAÇÃO PROFISSIONAL - EEEP

ENSINO MÉDIO INTEGRADO À EDUCAÇÃO PROFISSIONAL

CURSO TÉCNICO EM INFORMÁTICA

PROGRAMAÇÃO WEB
HTML - CSS - PHP

GOVERNO DO ESTADO DO CEARÁ

Secretaria da Educação

Governador
Cid Ferreira Gomes

Vice Governador
Francisco José Pinheiro

Secretária da Educação
Maria Izolda Cela de Arruda Coelho

Secretário Adjunto
Maurício Holanda Maia

Secretário Executivo
Antônio Idilvan de Lima Alencar

Assessora Institucional do Gabinete da Seduc
Cristiane Carvalho Holanda

Coordenadora de Desenvolvimento da Escola
Maria da Conceição Ávila de Misquita Vinãs

Coordenadora da Educação Profissional – SEDUC
Thereza Maria de Castro Paes Barreto

Índice

O que é Internet

Estamos acostumados a ouvir que Internet é a "grande rede mundial de computadores"

Entretanto, essa definição não é exata. Na realidade,

> *Internet é "o **conjunto** de diversas redes de computadores que se comunicam através dos protocolos TCP/IP"*

O hardware para conexão à Internet

Para conexão discada, é preciso ter um modem para comunicação com o provedor de acesso via linha telefônica comum.

Para conexão dedicada ADSL, é preciso ter uma placa de rede Ethernet 10/100 e um modem ADSL, além de um separador de sinais do telefone e da transmissão de dados.

Para conexão dedicada a cabo, é preciso um cablemodem e também um separador de sinais de TV e dos dados.

Para conexão dedicada wireless, é preciso um receptor de microondas e uma antena externa para o acesso à rede do provedor.

O software básico para conexão à Internet

Cada equipamento de hardware tem seus programas próprios para seu funcionamento.

Além do software associado ao hardware para conexão à Internet, é preciso ter pelo menos um browser (navegador) para se poder visitar os sites disponíveis.

Exemplos de browser (navegador): Netscape, Internet Explorer, Opera, Mozilla.

Protocolos

Assim como temos nossas regras sociais de comunicação (por exemplo, em uma palestra somente uma pessoa fala; em uma assembléia, são várias as pessoas que falam e, mesmo assim, uma pessoa fala por vez), também os computadores precisam de algumas regras para trocar informações. No caso da Internet, essas regras básicas estão reunidas no conjunto de protocolos chamados TCP/IP.

O protocolo IP

Na Internet cada computador tem um número IP próprio, assim como cada casa tem um endereço único.

Empresas que têm redes ligadas dia e noite na Internet possuem o que se chama *acesso dedicado*, isto é, as conexões de sua rede têm sempre um mesmo número IP na Internet.

Quando temos acesso à Internet através de um provedor, usamos o que se chama *acesso discado*, e nossa conexão com a Internet em geral ganha números IP diferentes a cada acesso. Mesmo assim, quando nosso computador se conecta ao provedor, o número IP atribuído a ele é único em toda a Internet.

O protocolo TCP

Suponhamos que em dado computador existem vários programas se comunicando através da rede em um mesmo instante - por exemplo, uma página da Web sendo carregada enquanto se verifica a caixa postal.

Como o computador "sabe" que a página da Web deve ir para o browser e os e-mails para o programa que lê e-mails?

Isso é possível porque cada programa em execução recebe também seu endereço próprio dentro do computador: no caso de programas que se comunicam pela Internet, esse endereço é o número TCP.

Assim, continuando a comparação com endereços físicos, suponhamos que seu computador é um prédio de apartamentos com um dado número IP; seu browser e seu programa de e-mail seriam apartamentos distintos nesse prédio, cada qual com seu número TCP.

Outros protocolos

Veremos que, para cada tipo de recurso disponível pela Internet, também existe um protocolo de comunicação específico, além do TCP/IP.

Recursos da Internet

E-mail

Serviço de intercâmbio assíncrono de mensagens, o "correio eletrônico" utiliza-se dos protocolos `POP` ou `IMAP` e `SMTP`.

FTP

Permite a transferência de arquivos, pelo protocolo `FTP` - *File Transfer Protocol*.

Telnet

Permite a conexão e interação com computadores remotos, simulando um terminal, pelo protocolo `telnet`.

Gopher

Sistema precursor da World-Wide Web, utiliza o protocolo `gopher` para apresentar menus de navegação, documentos e imagens.

O que é World-Wide Web

A World-Wide Web (também chamada Web ou WWW) é, termos gerais, ***a interface gráfica da Internet***. Ela é um sistema de informações organizado de maneira a englobar todos os outros sistemas de informação disponíveis na Internet.

Sua idéia básica é criar ***um mundo de informações sem fronteiras***, prevendo as seguintes características:

- interface consistente;
- incorporação de um vasto conjunto de tecnologias e tipos de documentos;
- "leitura universal".

Para isso, implementa três ferramentas importantes:

- um protocolo de transmissão de dados - HTTP;
- um sistema de endereçamento próprio - URL;
- uma linguagem de marcação, para transmitir documentos formatados através da rede - HTML.

HTTP

`HTTP` significa `HyperText Transfer Protocol` - *Protocolo de Transferência de Hipertexto*.

O `HTTP` é o protocolo usado para a transmissão de dados no sistema World-Wide Web. Cada vez que você aciona um link, seu *browser* realiza uma comunicação com um servidor da Web através deste protocolo.

URL

O sistema de endereçamento da Web é baseado em uma sintaxe chamada `URI` (`Universal Resource Identifier` - *Identificador Universal de Recursos*). Os endereços que utilizamos atualmente são os `URL`s, que seguem essa sintaxe.

`URL` significa `Uniform Resource Locator` - *Localizador Uniforme de Recursos*.

Um exemplo de `URL` é:

```
http://www.icmc.usp.br/ensino/material/html/url.html
```

Esse endereço identifica:

- ***o protocolo de acesso*** ao recurso desejado (`http`),
- ***a máquina*** a ser contactada (`www.icmc.usp.br`),
- ***o caminho de diretórios*** até o recurso (`ensino/material/html/`), e
- ***o recurso*** (arquivo) a ser obtido (`url.html`).

Através de `URL`s também acionamos programas (*scripts*), enviamos parâmetros para esses programas, etc.

HTML

`HTML` significa `HyperText Markup Language` - *Linguagem de Marcação de Hipertexto*.

Não é possível programar em linguagem `HTML`, pois ela é simplesmente uma linguagem de marcação: ela serve para indicarmos formatações para textos, inserir imagens e ligações de hipertexto.

Os *browsers* são os responsáveis por identificar as marcações em `HTML` e apresentar os documentos conforme o que foi especificado por essas marcações.

Introdução à Linguagem HTML

HTML *(HyperText Markup Language - Linguagem de Formatação de Hipertexto)* é fruto do "casamento" dos padrões HyTime e SGML.

HyTime - Hypermedia/Time-based Document Structuring Language

Hy Time (ISO 10744:1992) - padrão para representação estruturada de hipermídia e informação baseada em tempo. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (áudio, vídeo, etc.), conectados por webs ou hiperlinks.

O padrão HyTime é independente dos padrões de processamento de texto em geral. Ele fornece a base para a construção de sistemas hipertexto padronizados, consistindo de documentos que alicam os padrões de maneira particular

SGML - Standard Generalized Markup Language

Padrão ISO 8879 de formatação de textos: não foi desenvolvido para hipertexto, mas torna-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações.

SGML não é padrão aplicado de maneira padronizada: todos os produtos SGML têm seu próprio sistema para traduzir as etiquetas para um particular formatador de texto.

- **DTD - Document Type Definition** - define as regras de formatação para uma dada classe de documentos. Um DTD ou uma referência para um DTD deve estar contido em qualquer documento conforme o padrão SGML.

Portanto, HTML é definido segundo um DTD de SGML.

Todo documento HTML apresenta elementos entre parênteses angulares (< e >); esses elementos são as *etiquetas* (*tags*) de HTML, que são os comandos de formatação da linguagem. A maioria das etiquetas tem sua correspondente de fechamento:

```
<etiqueta>...</etiqueta>
```

Isso é necessário porque as etiquetas servem para definir a formatação de uma porção de texto, e assim marcamos onde começa e termina o texto com a formatação especificada por ela.

Alguns elementos são chamados “vazios”, pois não marcam uma região de texto, apenas inserem alguma coisa no documento:

```
<etiqueta>
```

Todos os elementos podem ter atributos:

```
<etiqueta atributo1=valor1 atributo2=valor2>...</etiqueta>
```

HTML é um recurso muito simples e acessível para a produção de documentos. Nestes “capítulos”, será possível aprender grande parte de seus elementos.

Publicação de documentos

Para que uma página esteja permanentemente disponível pela Web, ela precisa ter um endereço fixo, alojada em um ***servidor***.

Existem vários provedores de espaço (*hosting*) gratuitos e também os provedores de acesso geralmente oferecem espaço para os sites de seus assinantes. Sites com fins lucrativos geralmente são hospedados em provedores de espaço pagos.

Definida a hospedagem, basta enviar para o provedor os arquivos de seu site (via FTP ou por uma página de envio no próprio provedor de espaço) e suas páginas já estarão disponíveis para visitas.

Na rede do ICMC

Para ter sua página pessoal, é necessário ter uma área na rede. Tendo sua área, o primeiro passo é criar, *em seu diretório raiz*, um diretório de nome **WWW** (em letras maiúsculas).

A partir do momento da criação desse diretório WWW, o URL

`http://www.icmc.usp.br/~seulogin/`

- ou `http://www.grad.icmc.usp.br/~seulogin/`, no caso de alunos de graduação

passa a ser reconhecido pelo servidor. Nesse diretório WWW deve haver um arquivo **index.html**, que será a sua página principal.

A seguir, certifique-se de que sua área e o diretório WWW dentro dela estejam com permissão de leitura para "todos" (no ambiente UNIX, dê o comando `chmod 755 WWW`).

Feito isso, se você é aluno do ICMC faça o cadastro de sua página aqui.

Obs.: Estas diretivas se aplicam em particular à rede do ICMC; outros sistemas podem ter outras configurações específicas. ***Se você não é usuário do ICMC***, visite a página de seu provedor de acesso à Internet ou provedor de e-mail (webmail) e procure informações sobre hospedagem de páginas.

Documento básico e seus componentes

A estrutura de um documento HTML apresenta os seguintes componentes:

```
<HTML>
<HEAD><TITLE>Titulo do Documento</TITLE></HEAD>
<BODY>
texto,
imagem,
links,
...
</BODY>
</HTML>
```

As etiquetas HTML **não são sensíveis à caixa**. Traduzindo: tanto faz escrever `<HTML>`, `<Html>`, `<html>`, `<HtMl>`, ...

Os documentos se dividem em duas seções principais, que veremos a seguir.

A seção <HEAD>

`<HEAD>` contém informações sobre o documento. O elemento **`<TITLE>`**, por exemplo, define um título, que é mostrado no alto da janela do browser. Nesta página, por exemplo, está definido assim:

```
<HEAD><TITLE>A seção &lt;HEAD&gt; - Tutorial HTML do ICMC-USP</TITLE></HEAD>
```

Todo documento WWW deve ter um título; esse título é referenciado em buscas pela rede, dando uma identidade ao documento. Para ver na prática a importância do título, se você adicionar esta página aos seus Favoritos (Bookmarks).

Note que o título da página se tornou a âncora de atalho para ela. Por isso é sugerido que os títulos dos documentos sejam sugestivos, evitando-se títulos genéricos como ***"Introdução"***. O título também é bastante significativo para a listagem de uma página nos resultados de pesquisas nos catálogos da Internet.

Além do título, **`<HEAD>`** contém outras informações de importância para os robôs de pesquisa, indicadas nos campos **`<META>`**.

Campos <META>

Os campos <META> têm dois atributos principais:

- `NAME`, indicando um nome para a informação
- `HTTP-EQUIV`, que faz uma correspondência com campos de cabeçalho do protocolo HTTP; a informação desse campo pode ser lida pelos browsers, e provocar algumas ações.

```
<HEAD>
<TITLE>Título do Documento</TITLE>
<META NAME="nome" CONTENT="valor">
<META HTTP-EQUIV="nome" CONTENT="valor">
</HEAD>
```

Este documento, por exemplo, tem as seguintes informações:

```
<HEAD>
<META HTTP-EQUIV="content-type" CONTENT="text/html; charset=iso-8859-1">
<META HTTP-EQUIV="pragma" CONTENT="no-cache">
<TITLE>A seção &lt;HEAD&gt; - Tutorial HTML do ICMC-USP</TITLE>
<META NAME="Author" CONTENT="Maria Alice Soares de Castro - masc@icmc.usp.br">
```

```
<META NAME="Generator" CONTENT="Namo WebEditor v5.0">
<META NAME="Description" CONTENT="Manual de referência para webdesigners e
desenvolvedores de sites">
<META NAME="KeyWords" CONTENT="HTML, WWW, Webpublishing, Internet, Webdesign">
<LINK REL="stylesheet" HREF="folhatut.css">
</HEAD>
```

Alguns valores dos atributos **META NAME** são inseridos automaticamente por alguns editores, por exemplo: Generator e Author. Os campos **Description** e **KeyWords** ajudam a classificação da página em algumas ferramentas de busca. Essas informações não têm qualquer efeito na apresentação da página, mas servem como uma explicação ou documentação sobre as informações contidas nela.

Há poucos valores para **META HTTP-EQUIV** em uso. O mais comum é **content-type**, que indica o conjunto de caracteres usado na página: essa informação ajuda o browser a exibir corretamente os caracteres especiais que estiverem presentes no texto.

Um exemplo de uso comum do atributo HTTP-EQUIV é promover a mudança automática de páginas, atribuindo-lhe o valor Refresh.

```
<HEAD>
<TITLE> ... </TITLE>
<META HTTP-EQUIV="Refresh" CONTENT="segundos; URL= pagina.html">
</HEAD>
```

onde:

pagina.html
: é a página a ser carregada automaticamente

segundos
: é o número de segundos passados até que a página indicada seja carregada.

Como foi comentado no exemplo, o efeito é interessante, mas para que serve? Se não pensamos em uma finalidade útil para esse efeito, caímos na tentação de usá-lo “à toa”.

A aplicação mais utilizada é a atualização automática de um documento que, por exemplo, tenha uma foto produzida por uma câmara de vídeo: pode-se forçar, com o Refresh, a atualização dessa página, mostrando para o leitor sempre uma imagem mais atual de algum evento sendo focalizado pela câmara. Outra utilização é em “chats”, ou em páginas que desviem a navegação por documentos desenvolvidos para browsers avançados.

A seção <BODY>

Tudo que estiver contido em **<BODY>** será mostrado na janela principal do browser, sendo apresentado ao leitor. **<BODY>** pode conter cabeçalhos, parágrafos, listas, tabelas, links para outros documentos, imagens, formulários, animações, vídeos, sons e scripts embutidos.
Veja um documento básico em HTML.

Atributos de <BODY>

Através de atributos de <BODY>, podemos definir cores para os textos, links e para o fundo das páginas, bem como uma imagem de fundo (marca d’água):

```
<BODY BGCOLOR="#rrggbb" TEXT="#rrggbb" LINK="#rrggbb" ALINK="#rrggbb" VLINK="#rrggbb"
BACKGROUND="URL">
```

onde:

BGCOLOR
: cor de fundo (quando não é indicada, o browser irá mostrar uma cor padrão, geralmente o cinza ou branco; alguns editores poderão estabelecer o branco para o fundo da página)

TEXT

cor dos textos da página (padrão: preto)

LINK

cor dos links (padrão: azul)

ALINK

cor dos links, quando acionados (padrão: vermelho)

VLINK

cor dos links, depois de visitados (padrão: azul escuro ou roxo)

Seus valores são dados em hexadecimal, equivalentes a cores no padrão RGB (`Red, Green, Blue`). Existem tabelas de cores com esses valores, mas grande parte dos editores já oferece uma interface bem amigável através da qual escolhemos as cores desejadas, sem nos preocuparmos com números esdrúxulos tais como `#FF80A0`.

Browsers que seguem a definição de HTML 3.2 em diante, também aceitam 16 nomes de cores, tirados da paleta VGA do Windows - por exemplo, podemos escrever **`BGCOLOR=`**`"BLUE"`. Porém, browsers mais antigos não apresentarão as cores indicadas.

BACKGROUND

indica o URL da imagem a ser replicada no fundo da página, como uma marca d'água. Veja o exemplo de uma página cuja imagem de fundo é New!.

Para efeitos de design, é possível fixar a imagem de fundo, para que ela não se mova junto com o texto ao se rolar a página. Esse efeito não é padrão e funciona no Internet Explorer.

Cabeçalhos

Há seis níveis de cabeçalhos em HTML, de **`<H1>`** a **`<H6>`**:

```
<H1>Este é um cabeçalho de nível 1</H1><H2>Este é um cabeçalho de nível 2</H2>
<H3>Este é um cabeçalho de nível 3</H3><H4>Este é um cabeçalho de nível 4</H4>
<H5>Este é um cabeçalho de nível 5</H5><H6>Este é um cabeçalho de nível 6</H6>
```

Esses cabeçalhos são mostrados da seguinte forma:

Este é um cabeçalho de nível 1

Este é um cabeçalho de nível 2

Este é um cabeçalho de nível 3

Este é um cabeçalho de nível 4

Este é um cabeçalho de nível 5

Este é um cabeçalho de nível 6

Aninhamento de cabeçalhos

Os cabeçalhos não podem ser aninhados, isto é, a formatação:

```
<H2>Este é <H1>um cabeçalho de nível 1</H1> dentro de um cabeçalho de nível 2</H2>
```

pode produzir algum resultado próximo ao desejado:

Este é um cabeçalho de nível 1

Dentro de um cabeçalho de nível 2

mas o mais comum é que os browsers "entendam" essa formatação como sendo:

```
<H2>Este é</H2> <H1>um cabeçalho de nível 1</H1> dentro de um cabeçalho de nível 2</H2>
```

- ou seja, como se estivesse faltando uma etiqueta de fechamento de <H2> antes de <H1>, e faltando uma abertura de <H2> depois do fechamento de <H1>, oferecendo o seguinte resultado:

Este é um cabeçalho de nível 1

dentro de um cabeçalho de nível 2

Os editores WYSIWYG naturalmente não permitem o aninhamento de cabeçalhos.

Alinhamento

Os cabeçalhos têm atributos de alinhamento:

```
<H2 ALIGN=CENTER>Cabeçalho centralizado</H2>
```

Cabeçalho centralizado

```
<H3 ALIGN=RIGHT>Cabeçalho alinhado à direita</H3>
```

Cabeçalho alinhado à direita

```
<H4 ALIGN=LEFT>Cabeçalho alinhado à esquerda (default)</H4>
```

Cabeçalho alinhado à esquerda (default)

Separadores

Como vimos no primeiro exemplo, as quebras de linha do texto fonte não são significativas na apresentação de documentos em HTML. Para organizar os textos, precisamos de separadores, apresentados aqui.

Quebra de linha

Quando queremos mudar de linha, usamos o elemento **
**. Isso só é necessário quando queremos uma quebra de linha em determinado ponto, pois os browsers já quebram as linhas automaticamente para

apresentar os textos.
Com sucessivos **
**, podemos inserir diversas linhas em branco nos documentos. Esse elemento tem um atributo especial, que será apresentado no item sobre inserção de imagens.

Parágrafos

Para separar blocos de texto, usamos o elemento **<P>**:
```
Parágrafo 1;<P>Parágrafo 2.
```
que produz:

Parágrafo1;

Parágrafo2.

Combinando parágrafos e quebras de linha, temos:
```
Parágrafo 1;<br> linha 1 do parágrafo 1, <br>linha 2 do parágrafo 1.<P>Parágrafo 2;<br> linha 1 do parágrafo 2, <br>linha 2 do parágrafo 2.
```
O resultado da marcação acima é:

Parágrafo 1;
linha 1 do parágrafo 1,
linha 2 do parágrafo 1.

Parágrafo 2;
linha 1 do parágrafo 2,
linha 2 do parágrafo 2.

<P> tem atributo de alinhamento, como os cabeçalhos:
```
<P ALIGN=CENTER>Assim como os trens, as boas idéias às vezes chegam com atraso. <BR>(Giovani Guareschi)</P>
```

Assim como os trens, as boas idéias às vezes chegam com atraso. (Giovani Guareschi)

```
<P ALIGN=RIGHT>Como diz o provérbio chinês: “É melhor passar por ignorante uma vez do que permanecer ignorante para sempre”.</P>
```

Como diz o provérbio chinês: “É melhor passar por ignorante uma vez do que permanecer ignorante para sempre”.

```
<P ALIGN=LEFT>Este é o alinhamento padrão (default), e por isso não vou colocar nenhuma frase especial.</P>
```

Este é o alinhamento padrão (default), e por isso não vou colocar nenhuma frase especial.

Linha Horizontal

<HR> insere uma linha horizontal:
Erro! Indicador não definido.
Essa linha tem diversos atributos, oferecendo resultados diversos.

<HR **SIZE**=7> insere uma linha de largura 7 (pixels):
Erro! Indicador não definido.
<HR **WIDTH**=50%> insere uma linha que ocupa 50% do espaço horizontal disponível:
Erro! Indicador não definido.
<HR **WIDTH**=30% **ALIGN**=RIGHT **NOSHADE**> insere uma linha de comprimento 30% (do espaço horizontal disponível), alinhada à direita, sem efeito tridimensional:
Erro! Indicador não definido.
<HR **SIZE**=70 **WIDTH**=2 **ALIGN**=LEFT> insere uma linha de largura 70 (pixels), comprimento 2 (pixels), alinhada à esquerda (o Netscape, aparentemente, não aceita esta formatação de <HR>):

Listas em HTML

Há vários tipos de listas em HTML, sendo estas as mais usadas e corretamente apresentadas pelos browsers:

Listas de Definição

Estas listas são chamadas também “Listas de Glossário”, uma vez que têm o formato:

```
<DL>
<DT>termo a ser definido
<DD>definição
<DT>termo a ser definido
<DD>definição
</DL>
```

Que produz:

termo a ser definido
    definição
termo a ser definido
    definição

Este tipo de lista é muito utilizado para diversos efeitos de organização de páginas, por permitir a tabulação do texto. Um exemplo são os índices de navega_1ção presentes nas pági nas deste tutorial; outro exemplo é a lista composta abaixo:

```
<DL>
<DT>Imperadores do Brasil:
<DD>D. Pedro I
      <DL>
      <DD>Nome completo: Pedro de Alcântara Francisco Antônio João Carlos Xavier de
      Paula Miguel Rafael Joaquim José Gonzaga Pascoal Cipriano Serafim de Bragança e
      Bourbon
      </DL>
<DD>D. Pedro II
      <DL>
      <DD>Nome completo: Pedro de Alcântara João Carlos Leopoldo Salvador Bibiano
      Francisco Xavier de Paula Leocádio Miguel Gabriel Rafael Gonzaga
      </DL>
</DL>
```

Imperadores do Brasil:
    D. Pedro I
    Nome completo: Pedro de Alcântara Francisco Antônio João Carlos Xavier de Paula Miguel Rafael Joaquim José Gonzaga Pascoal Cipriano Serafim de Bragança e Bourbon
    D. Pedro II
    Nome completo: Pedro de Alcântara João Carlos Leopoldo Salvador Bibiano Francisco Xavier de Paula Leocádio Miguel Gabriel Rafael Gonzaga

Listas não-numeradas

São equivalentes às listas com marcadores do MS Word:

```
<UL>
<LI>item de uma lista
<LI>item de uma lista, que pode ser tão grande quanto se queira, sem que seja necessário se preocupar com a formatação das margens de texto
<LI>item
</UL>
```

- item de uma lista
- item de uma lista, que pode ser tão grande quanto se queira, sem que seja necessário se preocupar com a formatação das margens de texto
- item

A diferença entre o resultado da marcação HTML e do Word está na mudança dos marcadores, assinalando os diversos níveis de listas compostas:

```
<UL>
<LI>Documentos básicos
<LI>Documentos avançados
      <UL>
      <LI>formulários
      <UL>
      <LI>CGI
      </UL>
      <LI>contadores
      <LI>relógios
      </UL>
<LI>Detalhes sobre imagens
</UL>
```

- Documentos básicos
- Documentos avançados
    - formulários
        - CGI
    - contadores
    - relógios
- Detalhes sobre imagens

Essa lista pode ter marcadores diferentes, indicados através do atributo `TYPE`, que assume os valores `CIRCLE`, `SQUARE` e `DISC` (default):

```
<UL TYPE=CIRCLE>
<LI>um item
<LI>mais um item
</UL>
```

- um item
- mais um item

Cada item também pode ter seu atributo específico:

```
<UL>
<LI TYPE=DISC>um item
<LI TYPE=CIRCLE>mais um item
<LI TYPE=SQUARE>último item
</UL>
```

- um item

- mais um item
  - último item

Listas Numeradas

```
<OL>
<LI>item de uma lista numerada
<LI>item de uma lista numerada, que pode ser tão grande quanto se queira, sem que seja necessário se preocupar com a formatação das margens de texto
<LI>item de lista numerada
</OL>
```

1. item de uma lista numerada
2. item de uma lista numerada, que pode ser tão grande quanto se queira, sem que seja necessário se preocupar com a formatação das margens de texto
3. item de lista numerada

Estas listas não apresentam numeração em formato `1.1`, `1.2` etc., quando compostas:

1. Documentos básicos
2. Documentos avançados
    1. formulários
        1. CGI
        2. contadores
        3. relógios
    2. Detalhes sobre imagens

Porém, através do atributo `TYPE` (HTML 3.2), pode-se lidar com a numeração dos itens:

```
<OL TYPE=I>
<LI>Documentos básicos
<LI>Documentos avançados
      <OL TYPE=a>
      <LI >formulários
      <OL TYPE=i>
      <LI>CGI
      </OL>
      <LI>contadores
      <LI>relógios
      </OL>
<LI>Detalhes sobre imagens
</OL>
```

I. Documentos básicos

II. Documentos avançados

   a. formulários

      i. CGI

   b. contadores

   c. relógios

III. Detalhes sobre imagens

Ainda segundo HTML 3.2, o atributo `START` pode indicar o início da numeração da lista:

```
<OL START=4 TYPE=A>
<LI>um item
<LI>outro item
<LI>mais um item
</OL>
```

D. um item

E. outro item

F. mais um item

Listas e “sub-listas”

As listas podem ser aninhadas. Por exemplo:

```
<DL>
<DT>termo a ser definido
<DD>definição
   <OL>
   <LI>item de uma lista numerada
   <LI>item de uma lista numerada
      <UL>
      <LI>item de uma lista
      </UL>
   <LI>item de uma lista numerada
   </OL>
<DT>termo a ser definido
<DD>definição
</DL>
```

termo a ser definido
: definição

1. item de uma lista numerada
2. item de uma lista numerada
    - item de uma lista
3. item de uma lista numerada

termo a ser definido
: definição

Formatação de textos e caracteres

Há dois tipos de formatação em HTML: ***lógico*** e ***físico***. Os efeitos de apresentação na tela são os mesmos: o motivo da distinção entre eles se deve à idéia básica de independência entre especificação e apresentação.

Quando formatamos um trecho de texto como cabeçalho de nível 1, não explicitamos se esse tipo de cabeçalho deve ser em alguma fonte determinada, em um tamanho determinado, justificado à esquerda ou à direita, ou centralizado. Esses detalhes de apresentação são deixados para o browser - o dispositivo de apresentação do documento - que pode ser configurado de acordo com o leitor (usuário final).

Desse modo, além de facilitar enormemente o trabalho de quem escreve os documentos, a linguagem garante a uniformidade de apresentação de cabeçalhos, parágrafos, listas, etc.

A formatação lógica segue o significado lógico do texto marcado: um endereço de e-mail, uma citação etc. Sua apresentação final varia conforme o browser, podendo oferecer resultados mais ricos.

A formatação física especifica explicitamente o estilo que se quer para o texto: itálico, grifado etc. Sua apresentação final não sofre grandes variações.

Blocos de texto

HTML oferece as seguintes formatações de blocos de texto:

<PRE>

Apresenta o texto na mesma maneira em que foi digitado, mantendo quebras de linha e tabulações:

```
<pre>uma         linha   aqui,
outra ali,
   etc.</pre>
```

Resulta em:

```
uma      linha   aqui,
outra ali,
   etc.
```

Uma vez que <PRE> mantém o texto original, não se deve forçar espaços com essa marcação dentro de outra marcação que já apresente tabulações e espaços específicos.

<BLOCKQUOTE>

É usado para citações longas:

```
<blockquote>A massa do Sol arqueia o espaço-tempo de tal maneira que, ainda que a Terra siga uma trajetória reta no espaço-tempo quadridimensional, parece-nos que se desloca em órbita circular no espaço tridimensional.</blockquote> (Stephen W. Hawking, “Uma Breve História do Tempo”)
```

A massa do Sol arqueia o espaço-tempo de tal maneira que, ainda que a Terra siga uma trajetória reta no espaço-tempo quadridimensional, parece-nos que se desloca em órbita circular no espaço tridimensional.

(Stephen W. Hawking, “Uma Breve História do Tempo”)

<ADDRESS>

Usado para formatar endereços E-mail e referências a autores de documentos:

```
Envie críticas e sugestões para <address>webmaster@icmc.usp.br</address>
```

Envie críticas e sugestões para

webmaster@icmc.usp.br

Formatação de frases

Como visto anteriormente (em Formatação de Textos e Caracteres), HTML permite dois tipos de formatação: ***lógico*** e ***físico***; aqui veremos as formatações mais utilizadas:

Estilos Lógicos

<CITE>
Para títulos de livros, filmes, e citações curtas. Exemplo:
Assisti *Guerra nas Estrelas* umas oito vezes!
<CODE>
Para indicar trechos de código de programas. Exemplo:
`for (x=0); cl &&(!feof(stdin)); x++));`
<DFN>
Indica definição de uma palavra, em geral apresenta o texto em itálico. Exemplo:
CERN: *Centre d'Éstudes et Recherches Nucleaires*
<EM>
Ênfase, também normalmente apresentado em itálico. Exemplo:
É preciso pesquisar *muito* para encontrar o termo exato.
<KBD>
Indica uma entrada via teclado. Exemplo:
Para ler mensagens recebidas, digite `pine -i`
<SAMP>
Indica uma seqüência de caracteres, por exemplo uma mensagem de erro ou um resultado. Exemplo:
O resultado do primeiro applet é: `Hello, World!`
<STRONG>
Forte ênfase, mostrado normalmente em negrito. Exemplo:
Antes de enviar um e-mail, **confira o campo "Subject:"**!
<VAR>
Indica variáveis, ou valores que o usuário deverá escrever; geralmente mostrado em itálico. Exemplo:
No campo `Login`, escreva *guest*.

Estilos Físicos

<B>
Quando disponível no browser, é mostrado em **negrito** (em alguns browsers, pode aparecer sublinhado)
<I>
Itálico (em alguns casos, caracteres inclinados)
<TT>
Tipo `teletype` - fonte de espaçamento fixo.
<U>
Sublinhado; deve ser usado com cuidado, pois confunde-se com a apresentação de *links*.
<STRIKE> ou **<S>**
Frase ~~riscada~~.
<BIG>
Fonte um pouco maior.
<SMALL>
Fonte um pouco menor.
<SUB>
Frase em estilo $_{\text{índice}}$, como em H_2O.
<SUP>
Frase em estilo $^{\text{expoente}}$, como em Km^2.

Caracteres especiais

HTML permite que caracteres especiais sejam representados por seqüências de escape, indicadas por três partes: um `&` inicial, um número ou cadeia de caracteres correspondente ao caracter desejado, e um `;` final.

Quatro caracteres ASCII - `<`, `>`, e `&` têm significados especiais em HTML, e são usados dentro de documentos seguindo a correspondência:

Entidade	Caracter

<lt;	<
<gt;	>
<amp;	&

Outras sequências de escape suportam caracteres *ISO Latin1*. Aqui está uma tabela com os caracteres mais utilizados em Português:

Erro! Indicador não definido. **Erro! Indicador não definido.**

Como vemos, as sequências de escape são sensíveis à caixa. Os editores de HTML fazem essa tradução automaticamente.

Alguns editores, no entanto, mantêm a acentuação, sem usar as entidades de formatação. Quando isso acontece, deve-se inserir uma indicação do esquema de codificação *ISO Latin1*, escrevendo:

```
<HTML>
<HEAD>
<TITLE>...</TITLE>
<META HTTP-EQUIV="Content-Type"
      CONTENT="text/html; charset=ISO-8859-1">
</HEAD>
...
```

Existem alguns símbolos que vêm sendo incorporados ao conjunto de caracteres reconhecidos em HTML. Por exemplo, `&copy;`, que é o símbolo ©, `&reg;` para ®, e `&sect;` para §.

Também se pode usar seqüências com códigos ASCII, por exemplo:

```
&#191;Qué pasa, señor?
```

¿Qué pasa, señor?

Cores e fontes

Cores

As cores são introduzidas através do elemento `<FONT>`, usando o sistema RGB para cores (da mesma forma que vimos para cores de documentos):

```
<FONT COLOR="#rrggbb">Texto</FONT>
```

Assim, um trecho de texto pode ter uma cor diferente da definição geral de cores, feita através dos atributos de `<BODY>`.

Tamanho

A formatação

```
<FONT SIZE=tamanho_da_letra>Texto</FONT>
```

permite que o autor do documento altere o tamanho das letras em trechos específicos de texto. O tamanho básico dos textos é 3. Podemos indicar tamanhos relativos a esse, por exemplo:

```
<FONT SIZE=+2>Letra maior</FONT>
Letra normal
<FONT SIZE=-2>Letra menor</FONT>
```

Letra maior Letra normal Letra menor

Fontes

Uma evolução que permite a escolha da fonte para os textos, é o atributo `FACE`:

```
<FONT FACE="fonte_da_letra">Texto</FONT>
```

Por exemplo:

```
<FONT FACE="Verdana" COLOR="#0000AA">Fonte Verdana azul</FONT>
```
Fonte Verdana azul

```
<FONT FACE="Arial" COLOR="#00AA00">Fonte Arial verde</FONT>
```
Fonte Arial verde

```
<FONT FACE="Courier New" COLOR="#AA0000">Fonte Courier New vermelha</FONT>
Fonte Courier New vermelha
```

Marquee

É possível obter o efeito de animação de texto, através da formatação `<MARQUEE>`.

```
<MARQUEE BEHAVIOR=efeito>Texto</MARQUEE>
```

Atributos de largura e direção do efeito permitem diversas apresentações diferentes. Por exemplo (o efeito só é executado no Internet Explorer e em versões recentes do Netscape - e de maneiras diferentes):

```
<MARQUEE BEHAVIOR=SCROLL WIDTH=30%>Texto</MARQUEE>
<MARQUEE BEHAVIOR=SCROLL WIDTH=30%>Texto</MARQUEE>

<MARQUEE BEHAVIOR=SLIDE DIRECTION=RIGHT>Texto</MARQUEE>

<MARQUEE BEHAVIOR=SLIDE DIRECTION=LEFT>Texto</MARQUEE>
```

Ligações (uso de links)

Com HTML é possível fazermos ligações de uma região de texto (ou imagem) a um outro documento. Nestas páginas, temos visto exemplos dessas ligações: o browser destaca essas regiões e imagens do texto, indicando que são ligações de hipertexto - também chamadas *hypertext links* ou *hiperlinks* ou simplesmente *links*.

Para inserir um link em um documento, utilizamos a etiqueta **<A>**, da seguinte forma:

```
<A HREF = "arq_destino">âncora</A>
```

onde:

`arq_destino`
: é o URL do documento de destino;

`âncora`
: é o texto ou imagem que servirá de ligação hipertexto do documento sendo apresentado para o documento de destino.

Atributos

`<A>` tem vários atributos, utilizados de acordo com a ação associada ao link. Os mais usados são:

`HREF`
: Indica o arquivo de destino da ligação de hipertexto.

`TARGET`
: Indica o frame em que será carregado o `arq_destino`. Maiores detalhes na seção sobre frames.

`NAME`
: Marca um indicador, isto é, uma região de um documento como destino de uma ligação.

Caminhos (uso de links)

Os links podem estar indicados como caminhos relativos ou absolutos.

Caminho relativo

O caminho relativo pode ser usado sempre que queremos fazer referência a um documento armazenado no mesmo servidor do documento atual.
Através do campo de endereço do browser, vemos que este documento está localizado em um diretório `/ensino/material/html/` do servidor `www.icmc.usp.br`. Para escrevermos um link deste documento para o documento `doc2.html` no diretório `/ensino/material/html/exemplos/`, tudo que precisamos fazer é escrever:

```
Veja o <A HREF="exemplos/doc2.html">exemplo de caminho relativo</A>.
```

Da mesma forma, se quisermos um link deste documento para um outro que esteja em diretório diferente neste mesmo servidor, escrevemos, por exemplo:

```
<A HREF="/ensino/material/">Instituto de Ciências Matemáticas e de Computação - Material Didático</A>
```

que produz o link: Instituto de Ciências Matemáticas e de Computação - Material Didático

Para usar links com caminhos relativos é preciso, portanto, conhecer a estrutura do diretório do servidor no qual estamos trabalhando. Quando há alguma dúvida, o melhor é usar o caminho absoluto.

Caminho absoluto

Utilizamos o caminho absoluto quando desejamos referenciar um documento que esteja em outro servidor, por exemplo:

```
<A HREF="http://www.labes.icmc.usp.br/">Laboratório de Engenharia de Software</A>
```

que oferece um link para um documento no servidor WWW do Laboratório de Engenharia de Software:

Laboratório de Engenharia de Software

Com a mesma sintaxe, é possível escrever links para qualquer servidor de informações da Internet.

Indicadores (uso de links)

Como foi dito anteriormente, o atributo `NAME` permite indicar um trecho de documento como *ponto de chegada* de uma ligação hipertexto.

A formatação:

```
<A NAME="inicio">Indicadores (uso de links)</A>
```

faz com que a âncora **Indicadores (uso de links)** seja o destino de um link. Se escrevermos:

```
<A HREF="#inicio">Topo do documento</A>.
```

teremos uma ligação hipertexto para um trecho deste mesmo documento:

Topo do documento.

Da mesma forma, construímos links para trechos determinados de outros documentos, desde que saibamos quais trechos do documento destino estão marcados para ponto de chegada de um link.

Por exemplo:

```
São Carlos é um <A HREF= "/ambiente/saocarlos/histprog.html#polo">pólo de alta tecnologia</A>.
```

produz um link para um parágrafo marcado com `<A NAME="polo">` no arquivo `histprogr.html` sobre a cidade de São Carlos, no diretório **/ambiente/saocarlos/**:

São Carlos é um pólo de alta tecnologia.

Inserção de imagens

O elemento **`IMG`** insere imagens que são apresentadas junto com os textos. Um atributo `SRC` deve estar presente, da seguinte forma:

```
<IMG SRC="URL_imagem">
```

onde **`URL_imagem`** é o URL do arquivo que contém a imagem que se quer inserir; pode ser referenciada uma imagem que esteja em um outro servidor (o que, logicamente, não é conveniente).

Assim, escrevendo:

```
<IMG SRC = "/icones/newred.gif">
```

inserimos a figura 

no documento.

As imagens usadas na Web são armazenadas em arquivos com extensão `*.gif`, `*.xbm`, `*.jpg` (ou `*.jpeg`), `*.png`.

Atributos básicos de imagem

ALT

Indica um texto alternativo, descrevendo brevemente a imagem, que é apresentado no lugar da imagem nos browsers texto, ou quando se desabilita o carregamento de imagens em browsers gráficos. É recomendável que esteja sempre presente.

```
<IMG SRC="URL_imagem" ALT="descrição_da_imagem">
```

Dessa forma, `<IMG SRC="/icones/newred.gif" ALT="Novo!">` é apresentado nos browsers gráficos assim: New!e, nos browsers texto, assim: `[Novo!]`

WIDTH e HEIGHT

Atributos de dimensão da imagem, em pixels. Grande parte dos editores HTML coloca automaticamente os valores destes atributos, quando indicamos a inserção de uma imagem.

```
<IMG SRC="imagem" ALT="descrição" WIDTH="largura" HEIGHT="altura">
```

Uma das vantagens de se usar esses atributos é que o browser pode montar mais rapidamente as páginas, por saber de antemão o espaço que deverá ser reservado a elas.

BORDER

Quando uma frase é marcada como âncora de um link, ela se apresenta sublinhada; quando uma imagem faz as vezes de âncora, ganha uma borda que indica sua condição. Por exemplo: 
Porém, por questões de apresentação, nem sempre interessa termos essa borda ao redor da imagem. Assim, com o atributo `BORDER`, podemos controlar esse detalhe.
Se quisermos uma borda mais larga... 

```
<A HREF="URL"><IMG SRC="imagem" ALT="descrição" BORDER=4></A>
```

Se quisermos uma imagem sem borda... 

```
<A HREF="URL"><IMG SRC="imagem" ALT="descrição" BORDER=0></A>
```

Essa borda pode ser apresentada também em imagens que não são âncora de links. Basta aplicar, por exemplo, a formatação

```
<IMG SRC="icones/fotoicm.gif" ALT="Foto antiga do ICMC" BORDER=2>
```

Assim, é possível dar mais destaque a uma imagem, sem ser necessário editá-la:

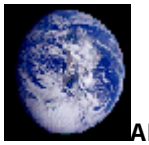
Foto original

Foto com borda gerada por HTML

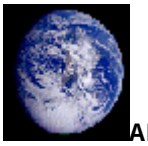
ALIGN

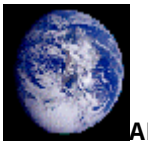
```
<IMG SRC="imagem" ALT="descrição" ALIGN=alinhamento>
```

Existem também atributos de alinhamento, que produzem os seguintes resultados:



ALIGN=TOP Alinha o texto adjacente com o topo da imagem, embora com linhas compridas o resultado não seja muito bom.

 **ALIGN=MIDDLE** Alinha o texto adjacente com o meio da imagem, embora com linhas compridas o resultado não seja muito bom

 **ALIGN=BOTTOM** Alinha o texto adjacente com a parte de baixo da imagem *(default)*

ALIGN=RIGHT Alinha imagem à direita, e tudo o que houver ao redor (texto, outras imagens) a partir do topo da imagem.

ALIGN=LEFT Alinha imagem à esquerda, e tudo o que houver ao redor (texto, outras imagens) a partir do topo da imagem.

Para ter duas imagens, uma em cada margem, numa mesma linha, escreva:

```
<IMG align=left SRC="imagem.gif" alt="imagem"><IMG align=right SRC="imagem.gif" alt="imagem">...e se pode escrever à vontade entre as imagens!
```

Um detalhe surgido com o alinhamento de imagens foi a necessidade de se liberar o texto desse alinhamento. Ou seja:
Suponhamos um texto mais ou menos curto, que desejamos colocar aqui, com a imagem ilustrativa...
...mas gostaríamos que **este** trecho já estivesse abaixo da imagem! De acordo com o comprimento da primeira frase, não seria possível usar o alinhamento `TOP`.
Para conseguir isso, seria necessário incluir diversos `<BR>` consecutivos, inserindo linhas em branco; mesmo assim, o resultado final poderia ser bem pouco elegante. Surgiu, então, o atributo `CLEAR` para `<BR>`.
Com esse atributo, podemos, por exemplo...
...ter um texto posicionado no ponto em que a margem direita fica livre, com `<BR CLEAR=RIGHT>`
ou no ponto em que a margem esquerda fica livre, com `<BR CLEAR=LEFT>`
Dessa maneira, podemos controlar bem a posição relativa dos textos.
Também se pode posicionar o texto no ponto em que ambas as margens estão livres. Isso é conseguido com `<BR CLEAR=ALL>`
E, assim, vimos tudo sobre quebras de linha depois de imagens!

Molduras de imagem

Para melhorar ainda mais a apresentação das imagens junto com os textos, foram desenvolvidos atributos de moldura. Estes atributos definem o espaço - vertical e horizontal - deixado entre as imagens e os textos circundantes:

```
<IMG SRC="imagem" VSPACE=espaço_vertical>
<IMG SRC="imagem" HSPACE=espaço_horizontal>
```

O efeito desses atributos pode ser percebido nos textos abaixo. No primeiro texto, as imagens não têm atributos de moldura (é fácil notar como o texto fica "grudado" na imagem)

O *Instituto de Ciências Matemáticas e de Computação* (ICMC-USP) é formado pelos Departamentos de Matemática e de Ciências de Computação e Estatística. O ICMC originou-se em 1953, como Departamento de Matemática da *Escola de Engenharia de São Carlos* (EESC-USP), fundado por renomados matemáticos italianos e brasileiros. Atualmente, o Departamento de Matemática oferece

cursos de Licenciatura e Bacharelado em Matemática em nível de graduação, além de um programa de pós-graduação que inclui mestrado e doutorado na área de Matemática. O Departamento de Computação e Estatística é responsável pelo curso de Bacharelado em Ciência de Computação, no qual ingressam 40 alunos por ano. Em nível de pós-graduação oferece, desde 1975, o programa de mestrado em Ciências de Computação e Matemática Computacional e, a partir de agosto de 1995, o programa de doutorado na mesma área.

Neste segundo texto são usadas, respectivamente, as formatações:

```
<IMG SRC="icones/fotoicm.gif" WIDTH="148" HEIGHT="95" ALIGN=left VSPACE="30">
e
<IMG SRC="icones/smallpos.gif" WIDTH="160" HEIGHT="71" ALIGN=right HSPACE="30">
```

O *Instituto de Ciências Matemáticas e de Computação* (ICMC-USP) é formado pelos Departamentos de Matemática e de Ciências de Computação e Estatística. O ICMC originou-se em 1953, como Departamento de Matemática da *Escola de Engenharia de São Carlos* (EESC-USP), fundado por renomados matemáticos italianos e brasileiros. Atualmente, o Departamento de Matemática oferece cursos de Licenciatura e Bacharelado em Matemática em nível de graduação, além de um programa de pós-graduação que inclui mestrado e doutorado na área de Matemática. O Departamento de Computação e Estatística é responsável pelo curso de Bacharelado em Ciência de Computação, no qual ingressam 40 alunos por ano. Em nível de pós-graduação oferece, desde 1975, o programa de mestrado em Ciências de Computação e Matemática Computacional e, a partir de agosto de 1995, o programa de doutorado na mesma área.

Os dois atributos de moldura podem estar presentes ao mesmo tempo. Vejamos primeiro o texto com a imagem sem moldura:

"A cultura UNIX começou a ser apreciada por usuários brasileiros ainda na década de 70, pelos contatos de pesquisadores brasileiros em cursos de aperfeiçoamento no exterior - notadamente na América do Norte. O contingente era, contudo, pequeno e restrito a acadêmicos. A disseminação da cultura UNIX no mercado comercial só teve início com o advento da década de 80."

(Citação de texto encontrado à página 18 do livro *UNIX - Guia do Usuário* - Autores: Marcus C. Sampaio, Jacques P. Sauvé e J. Antão B. Moura - McGraw-Hill, 1988)

Abaixo, vemos a aplicação dos dois atributos, através da formatação:

```
<IMG SRC="icones/earth.gif" ALIGN="LEFT" WIDTH="63" HEIGHT="68" HSPACE="20" VSPACE="20">
```

"A cultura UNIX começou a ser apreciada por usuários brasileiros ainda na década de 70, pelos contatos de pesquisadores brasileiros em cursos de aperfeiçoamento no exterior - notadamente na América do Norte. O contingente era, contudo, pequeno e restrito a acadêmicos. A disseminação da cultura UNIX no mercado comercial só teve início com o advento da década de 80."

(Citação de texto encontrado à página 18 do livro *UNIX - Guia do Usuário* - Autores: Marcus C. Sampaio, Jacques P. Sauvé e J. Antão B. Moura - McGraw-Hill, 1988)

Tabelas

A formatação de tabelas foi adotada bem antes de sua inclusão na definição de HTML. A manipulação de tabelas, mesmo em editores, é trabalhosa; a maior diferença entre tabelas em HTML e em editores como o MS Word, entretanto, é o fato das tabelas em HTML serem definidas apenas *em termos de linhas* e não de colunas. Mas isso será percebido no decorrer destas páginas.

As tabelas foram uma grande conquista para os autores de documentos para a Web. Com elas é possível, por exemplo, termos estas páginas do tutorial organizadas em colunas, sendo uma delas reservada aos

links de navegação dentro de cada seção.

Tabelas implementam um conceito importante de *layout*: as “grades”, segundo as quais organizamos textos e ilustrações de maneira harmoniosa.

Como já foi possível perceber, as tabelas contêm textos, listas, parágrafos, imagens, formulários e várias outras formatações - inclusive outras tabelas. Novas versões de HTML e de browsers populares vêm acrescentando diversos atributos às tabelas, e nosso objetivo aqui é saber lidar com a maioria desses recursos disponíveis.

Elementos básicos de tabelas

`<TABLE>...</TABLE>` delimita uma tabela. Um atributo básico é **`BORDER`**, que indica a apresentação da borda.

```
<TABLE BORDER="borda">
...
</TABLE>
```

Títulos, linhas e elementos

`<CAPTION>...</CAPTION>`
: define o título da tabela

`<TR>...</TR>`
: delimita uma linha

`<TH>...</TH>`
: define um cabeçalho para colunas ou linhas (dentro de `<TR>`)

`<TD>...</TD>`
: delimita um elemento ou célula (dentro de `<TR>`)

Uma tabela simples:

```
<TABLE BORDER=4>
<CAPTION>Primeiro exemplo</CAPTION>
<TR><TH>Coluna 1</TH><TH>Coluna 2</TH></TR>
<TR><TD>linha1, coluna 1</TD><TD> linha 1, coluna 2</TD></TR>
<TR><TD>linha 2, coluna 1</TD><TD>linha 2, coluna 2</TD></TR>
</TABLE>
```

Primeiro exemplo

Coluna 1	**Coluna 2**
linha1, coluna 1	linha 1, coluna 2
linha 2, coluna 1	linha 2, coluna 2

Títulos compreendendo mais de uma coluna ou linha

É possível englobar colunas e linhas, através dos atributos **`COLSPAN`** (para colunas) e **`ROWSPAN`** (para linhas):

```
<TABLE BORDER=1>
<TR><TH COLSPAN=2>Colunas 1 e 2</TH></TR>
<TR><TD>linha1, coluna 1</TD><TD> linha 1, coluna 2</TD></TR>
<TR><TD>linha 2, coluna 1</TD><TD>linha 2, coluna 2</TD></TR>
<TR><TH ROWSPAN=3>3 linhas</TH><TD>uma linha</TD></TR>
<TR><TD>duas linhas</TD></TR>
<TR><TD>tres linhas</TD></TR>
</TABLE>
```

Colunas 1 e 2	
linha1, coluna 1	linha 1, coluna 2
linha 2, coluna 1	linha 2, coluna 2

3 linhas	uma linha
	duas linhas
	tres linhas

Neste exemplo, vemos que o cabeçalho **Colunas 1 e 2** compreende duas colunas (`COLSPAN=2`); o cabeçalho **3 linhas** compreende, por sua vez, 3 linhas (`ROWSPAN=3`).

Tabelas sem borda

As páginas deste tutorial foram construídas com tabelas sem borda. Para tanto, foi empregada a seguinte delaração:

```
<TABLE BORDER="0">
...
</TABLE>
```

Alinhamentos em tabelas

Este exemplo servirá para estudarmos alinhamentos, controle de larguras e espaçamento em tabelas:

 **Prédio principal do ICMC-USP**	O *Instituto de Ciências Matemáticas e de Computação* (ICMC-USP) é formado pelos Departamentos de Matemática e de Ciências de Computação e Estatística. O ICMC originou-se em 1953, como Departamento de Matemática da *Escola de Engenharia de São Carlos* (EESC-USP), fundado por renomados matemáticos italianos e brasileiros.
Departamento de Matemática (SMA)	Atualmente, o Departamento de Matemática oferece cursos de Licenciatura e Bacharelado em Matemática e de Bacharelado em Matemática Aplicada e Computação Científica.
Departamento de Computação e Estatística (SCE)	O Departamento de Computação e Estatística é responsável pelo Bacharelado em Ciência de Computação e pelo curso noturno de Bacharelado em Informática.
Para maiores informações: Cursos de Graduação: grad@icmc.usp.br Cursos de Pós-Graduação: posgrad@icmc.usp.br	

O conteúdo é informativo, porém a apresentação não é agradável devido à disposição do texto na tabela. Primeiro, vamos mexer com os alinhamentos.

Alinhamentos simples

Os alinhamentos padrão em tabelas, como podemos ver no exemplo acima, são:

no sentido horizontal: alinhamento à esquerda
no sentido vertical: alinhamento no centro da célula

As linhas e células podem ter alinhamentos definidos através dos atributos:

`ALIGN` = alin_horizontal
`VALIGN` = alin_vertical

Vejamos como esses alinhamentos funcionam nas células:

```
<TD ALIGN=alin_horizontal>Texto da célula</TD>
<TD VALIGN=alin_vertical>Texto da célula</TD>
```

Padrão	**ALIGN**=LEFT	**ALIGN**=CENTER	**ALIGN**=RIGHT
Padrão	**VALIGN**=TOP	**VALIGN**=MIDDLE	**VALIGN**=BOTTOM

Obs.: a tabela acima foi feita especialmente para mostrar as diferenças entre os alinhamentos. Uma tabela comum ajusta o tamanho de suas células ao conteúdo, desta forma:

Padrão	align=left	align=center	align=right
Padrão	valign=top	valign=middle	valign=bottom

Alinhamentos combinados

Uma mesma célula pode ter atributos `ALIGN` e `VALIGN`:

```
<TD ALIGN=alin_horizontal VALIGN=alin_vertical>Texto da célula</TD>
```

Por exemplo:

Padrão	**ALIGN**=LEFT, **VALIGN**=BOTTOM	**ALIGN**=CENTER, **VALIGN**=TOP	**ALIGN**=RIGHT, **VALIGN**=MIDDLE

Alinhamentos de linhas

O alinhamento pode ser aplicado a linhas inteiras, com:

```
<TR ALIGN=alin_horizontal VALIGN=alin_vertical>Texto da célula</TR>
```

Porém, o alinhamento declarado em uma célula prevalece sobre o alinhamento da linha, como se vê no exemplo:

center	center	center	TD **ALIGN**=RIGHT
bottom	TD **VALIGN**=TOP	bottom	bottom

Isso pode ser interessante para algumas aplicações.

Já conseguimos mexer um pouco na tabela inicial, inserindo alinhamentos combinados; serão necessários mais alguns passos para que a tabela fique realmente "apresentável" - o exemplo continua nos itens sobre larguras e espaçamentos.

 **Prédio principal do ICMC-USP**	O *Instituto de Ciências Matemáticas e de Computação* (ICMC-USP) é formado pelos Departamentos de Matemática e de Ciências de Computação e Estatística. O ICMC originou-se em 1953, como Departamento de Matemática da *Escola de Engenharia de São Carlos* (EESC-USP), fundado por

	renomados matemáticos italianos e brasileiros.
Departamento de Matemática (SMA)	Atualmente, o Departamento de Matemática oferece cursos de Licenciatura e Bacharelado em Matemática e de Bacharelado em Matemática Aplicada e Computação Científica.
Departamento de Computação e Estatística (SCE)	O Departamento de Computação e Estatística é responsável pelo Bacharelado em Ciência de Computação e pelo curso noturno de Bacharelado em Informática.

Atributos de largura

No item anterior, foi comentado que uma tabela comum ajusta o tamanho de suas células ao conteúdo. Por exemplo:

janeiro	fevereiro	março
abril	maio	junho

Para apresentar uma tabela ocupando determinado espaço disponível na linha, usamos o atributo `WIDTH`. Esse atributo pode ser aplicado também a linhas e células.

Essa largura pode ser definida em porcentagem (do espaço disponível):

```
WIDTH=x%
```

ou em pixels:

```
WIDTH=x
```

Ex.1: Tabela ocupando 50% do espaço disponível

```
<TABLE BORDER=1 width=50%>
```

janeiro	fevereiro	março
abril	maio	junho

Ex.2: Tabela ocupando 50% do espaço disponível, com uma coluna de 60% do espaço disponível na tabela

```
<TABLE BORDER=1 width=50%>
<TR>
      <TD>janeiro</TD><TD width=60%>fevereiro</TD><TD>março</TD>
</TR>
<TR>
      <TD>abril</TD><TD width=60%>maio</TD><TD>junho</TD>
</TR>
</TABLE>
```

janeiro	fevereiro	março
abril	maio	junho

Ex3.: O controle da largura da tabela está limitado à dimensão de seu conteúdo:

```
<TABLE BORDER=1 width=50%>
<TR>
      <TD>janeiro</TD><TD width=1%>fevereiro</TD><TD>março</TD>
</TR>
<TR>
      <TD>abril</TD><TD width=1%>maio</TD><TD>junho</TD>
</TR>
</TABLE>
```

janeiro	fevereiro	março

abril	maio	junho

De volta ao exemplo inicial, já podemos melhorar um pouco mais nossa tabela. Mantendo os alinhamentos definidos na seção anterior, aplicaremos atributos de largura:

 **Prédio principal do ICMSC-USP**	O *Instituto de Ciências Matemáticas e de Computação* (ICMC-USP) é formado pelos Departamentos de Matemática e de Ciências de Computação e Estatística. O ICMC originou-se em 1953, como Departamento de Matemática da *Escola de Engenharia de São Carlos* (EESC-USP), fundado por renomados matemáticos italianos e brasileiros.
Departamento de Matemática (SMA)	Atualmente, o Departamento de Matemática oferece cursos de Licenciatura e Bacharelado em Matemática e o Bacharelado em Matemática Aplicada e Computação Científica.
Departamento de Computação e Estatística (SCE)	O Departamento de Computação e Estatística é responsável pelo Bacharelado em Ciência de Computação e o curso noturno de Bacharelado em Informática.

Atributos de espaçamento

Dois atributos permitem o controle de espaçamento em tabelas:

`CELLPADDING` - espaço entre o texto e as bordas da célula

`CELLSPACING` - espaço entre células

Tomemos a mesma tabela simples da seção anterior:

janeiro	fevereiro	março
abril	maio	junho

Ex.1: Espaço entre o texto e as bordas

```
<TABLE BORDER=1 CELLPADDING=20>
```

janeiro	fevereiro	março
abril	maio	junho

Ex.2: Espaço entre células

```
<TABLE BORDER=1 CELLSPACING=20>
```

janeiro	fevereiro	março
abril	maio	junho

Ex3.: Espaço entre texto e bordas, e espaço entre células

```
<TABLE BORDER=1 CELLPADDING=20 CELLSPACING=20>
```

janeiro	fevereiro	março

abril	maio	junho

Assim, damos mais uma mexida na tabela inicial:

 **Prédio principal do ICMSC-USP**	O *Instituto de Ciências Matemáticas e de Computação* (ICMC-USP) é formado pelos Departamentos de Matemática e de Ciências de Computação e Estatística. O ICMC originou-se em 1953, como Departamento de Matemática da *Escola de Engenharia de São Carlos* (EESC-USP), fundado por renomados matemáticos italianos e brasileiros.
Departamento de Matemática (SMA)	Atualmente, o Departamento de Matemática oferece cursos de Licenciatura e Bacharelado em Matemática e o Bacharelado em Matemática Aplicada e Computação Científica.
Departamento de Computação e Estatística (SCE)	O Departamento de Computação e Estatística é responsável pelo Bacharelado em Ciência de Computação e o curso noturno de Bacharelado em Informática.
Para maiores informações: Cursos de Graduação: grad@icmsc.sc.usp.br Cursos de Pós-Graduação: posgrad@icmsc.sc.usp.br	

Como toque final, retiramos a borda:

Prédio principal do ICMC-USP

O *Instituto de Ciências Matemáticas e de Computação* (ICMC-USP) é formado pelos Departamentos de Matemática e de Ciências de Computação e Estatística.

O ICMC originou-se em 1953, como Departamento de Matemática da *Escola de Engenharia de São Carlos* (EESC-USP), fundado por renomados matemáticos italianos e brasileiros.

Departamento de Matemática (SMA)

Atualmente, o Departamento de Matemática oferece cursos de Licenciatura e Bacharelado em Matemática e o Bacharelado em Matemática Aplicada e Computação Científica.

Departamento de Computação e Estatística (SCE)

O Departamento de Computação e Estatística é responsável pelo Bacharelado em Ciência de Computação e o curso noturno de Bacharelado em Informática.

Para maiores informações:

Cursos de Graduação: grad@icmc.usp.br

Cursos de Pós-Graduação: posgrad@icmc.usp.br

Agora já vimos grande parte dos recursos disponíveis para manipular tabelas, que permitem produzir bons efeitos de apresentação.

Extensões de tabelas

Diversas extensões de tabelas possibilitam a apresentação de efeitos muito bons nas páginas.

Cor de fundo

```
<TABLE BORDER=5 CELLSPACING=5 CELLPADDING=10 BGCOLOR="#E1FFD9">
```

janeiro	fevereiro	março
abril	maio	junho

```
<TABLE BORDER=5 CELLSPACING=5 CELLPADDING=10>
<TR><TD BGCOLOR="#E1FFD9">janeiro</TD><TD>fevereiro</TD>
<TD BGCOLOR="#E1FFD9">março</TD></TR>
<TR><TD>abril</TD><TD BGCOLOR="#E1FFD9">maio</TD><TD>junho</TD></TR>
</TABLE>
```

janeiro	fevereiro	março
abril	maio	junho

Cor de borda

```
<TABLE BORDER=5 CELLSPACING=5 CELLPADDING=10 BGCOLOR="#E1FFD9" BORDERCOLOR="#00FF00">
```

janeiro	fevereiro	março
Abril	maio	junho

```
<TABLE BORDER="1" CELLSPACING="0" CELLPADDING=10 BORDERCOLOR="#00FF00">
<TR>
<TD bgcolor="#E1FFD9">janeiro</TD><TD>fevereiro</TD>
<TD bgcolor="#E1FFD9">março</TD>
</TR>
<TR>
<TD>abril</TD><TD bgcolor="#E1FFD9">maio</TD><TD>junho</TD>
</TR>
</TABLE>
```

janeiro	fevereiro	março
abril	maio	junho

Imagem de fundo

```
<TABLE BORDER=5 BACKGROUND="imagem">
```

janeiro	fevereiro	março
abril	maio	junho

```
<TD BACKGROUND="imagem">
```

janeiro	fevereiro	março
abril	maio	junho

Frames

Os frames são divisões da tela do browser em diversas telas (ou “quadros”). Com isso, torna-se possível apresentar mais de uma página por vez: por exemplo, um índice principal em uma parte pequena da tela, e os textos relacionados ao índice em outra parte.

É muito fácil colocar frames em páginas; porém, nem todos os usuários gostam deles, pois nem sempre a navegação é fácil, além de problemas para a impressão e a marcação dos documentos interiores aos frames nos bookmarks. A alternativa natural para os frames são as tabelas.

Uma página com frames tem um texto fonte semelhante a:

```
<HTML>
<HEAD><TITLE>Assunto X</TITLE></HEAD>
<FRAMESET COLS="20%, 80%">
      <FRAME SRC="indice1.html">
      <FRAME SRC="apresenta.html" NAME="principal">
      <NOFRAME>
      <BODY>
      <H2>Bem-vindo à página do assunto X!</h2>
      <P>
      Blá blá blá blá blá
      blá blá blá blá blá
      </BODY>
      </NOFRAME>
</FRAMESET>
</HTML>
```

A parte **FRAMESET** define a divisão da página em “quadros”. Neste exemplo, a página será dividida em duas colunas, sendo a primeira com 20% do tamanho da tela, e a segunda coluna com os restantes 80% da tela.

Dentro da formatação de **FRAMESET**, temos os **FRAME SRC**, que são referências às páginas que serão mostradas nos frames definidos

Assim, no código acima vemos que a página `indice1.html` será mostrada na primeira coluna (que ocupará 20% da tela), e a página `apresenta.html` será mostrada na segunda (ocupando 80% da tela).

A formatação de frames inclui também uma parte **NOFRAME**, que é mostrada normalmente pelos browsers que não suportam sua apresentação.

Atributos de Frames

Até este ponto, vimos os atributos **COLS** e **ROWS** (para `FRAMESET`), **SRC** e **NAME** (para `FRAME`).

Outros atributos permitem um maior controle sobre a apresentação:

Eliminação das bordas dos frames:

```
<FRAMESET COLS="20%, 80%" FRAMEBORDER="no">
      <FRAME SRC="indice4.html">
      <FRAME SRC="apresenta4.html" NAME="principal">
</FRAMESET>
```

Frame sem barra de rolagem:

```
<FRAMESET COLS="20%, 80%">
      <FRAME SRC="indice4.html" SCROLLING="no">
      <FRAME SRC="apresenta4.html" NAME="principal">
</FRAMESET>
```

[Veja o exemplo]

É bom lembrar que a barra de rolagem de um frame fica sempre à direita; não é possível, atualmente, mudar essa característica.

Aplicações de Frames e Cuidados

Frames são interessantes para apresentar conjuntos de páginas com um índice fixo para a navegação.

Além disso, torna-se possível mostrar diversas páginas e/ou mídias em uma única janela do browser.

Um cuidado é procurar controlar bem a navegação, evitando que o acionamento de links não leve o leitor a ver seu browser criar frames dentro de frames, gerando uma grande confusão (veja item seguinte, sobre "limpar" a tela).

Limpando a tela

Há basicamente dois efeitos possíveis para limpar a apresentação de frames, e isso é feito com "targets" especiais (o atributo `TARGET` foi apresentado no item Links com frames):

`TARGET="_top"` limpa os frames que estiverem ativos, apresentando a página de destino na tela inteira

`TARGET="_blank"` abre uma nova janela do browser para apresentar a página de destino

Interação

A interação é realizada de duas formas diferentes:

1. através de programas executados/interpretados pelo browser (isto é, do lado do *cliente*);
2. através de programas executados pelo *servidor* HTTP.

Conforme a aplicação, apenas um destes tipos de interação pode ou deve ser utilizado.

Formulários

Um formulário é um modelo para a entrada de um conjunto de dados. O primeiro passo para fazer formulários é aprender as etiquetas que desenham as janelinhas de entrada de dados, para depois trabalharmos com os *scripts*, que são os programas que tratam esses dados, oferecendo os serviços desejados (acesso a banco de dados, envio de e-mail, etc.).

O elemento `<FORM>` delimita um formulário e contém uma seqüência de elementos de entrada e de formatação do documento.

```
<FORM ACTION="URL_de_script" METHOD=método>...</FORM>
```

Os atributos de `FORM` que nos interessam agora são:

`ACTION`
: Especifica o `URL` do *script* ao qual serão enviados os dados do formulário.

`METHOD`
: Seleciona um método para acessar o URL de ação. Os métodos usados atualmente são `GET` e `POST`. Ambos os métodos transferem dados do browser para o servidor, com a seguinte diferença básica:

- `POST`

  - os dados entrados fazem parte do corpo da mensagem enviada para o servidor;
  - transfere grande quantidade de dados.

- `GET`

  - os dados entrados fazem parte do URL (endereço) associado à consulta enviada para o servidor;
  - suporta até 128 caracteres.

Veremos maiores detalhes sobre métodos no item `CGI`.

`FORM` também pode apresentar o atributo:

`ENCTYPE`

Indica o tipo de codificação dos dados enviados através do formulário. O tipo *default* é `application/x-www-form-urlencoded`. Outro tipo aceito por alguns browsers é `text/plain`.

Os formulários podem conter qualquer formatação - parágrafos, listas, tabelas, imagens - exceto outros formulários. Em especial, colocamos dentro da marcação de `<FORM>` as formatações para campos de entrada de dados, que são três: `<INPUT>`, `<SELECT>` e `<TEXTAREA>`.

Todos os campos de entrada de dados têm um atributo `NAME`, ao qual associamos um nome, que será utilizado posteriormente pelo *script*. São os *scripts* que organizam esses dados de entrada em um conjunto de informações significativas para determinado propósito.

Primeiro vamos ver os tipos de campos para montar um formulário, e depois passaremos aos *scripts*.

Formulários

Um formulário é um modelo para a entrada de um conjunto de dados. O primeiro passo para fazer formulários é aprender as etiquetas que desenham as janelinhas de entrada de dados, para depois trabalharmos com os *scripts*, que são os programas que tratam esses dados, oferecendo os serviços desejados (acesso a banco de dados, envio de e-mail, etc.).

O elemento `<FORM>` delimita um formulário e contém uma seqüência de elementos de entrada e de formatação do documento.

```
<FORM ACTION="URL_de_script" METHOD=método>...</FORM>
```

Os atributos de `FORM` que nos interessam agora são:

`ACTION`
Especifica o `URL` do *script* ao qual serão enviados os dados do formulário.

`METHOD`
Seleciona um método para acessar o URL de ação. Os métodos usados atualmente são `GET` e `POST`. Ambos os métodos transferem dados do browser para o servidor, com a seguinte diferença básica:

- `POST`

  - os dados entrados fazem parte do corpo da mensagem enviada para o servidor;
  - transfere grande quantidade de dados.

- `GET`

  - os dados entrados fazem parte do URL (endereço) associado à consulta enviada para o servidor;
  - suporta até 128 caracteres.

Veremos maiores detalhes sobre métodos no item `CGI`.

`FORM` também pode apresentar o atributo:

`ENCTYPE`
Indica o tipo de codificação dos dados enviados através do formulário. O tipo *default* é `application/x-www-form-urlencoded`. Outro tipo aceito por alguns browsers é `text/plain`.

Os formulários podem conter qualquer formatação - parágrafos, listas, tabelas, imagens - exceto outros formulários. Em especial, colocamos dentro da marcação de `<FORM>` as formatações para campos de entrada de dados, que são três: `<INPUT>`, `<SELECT>` e `<TEXTAREA>`.

Todos os campos de entrada de dados têm um atributo `NAME`, ao qual associamos um nome, que será utilizado posteriormente pelo *script*. São os *scripts* que organizam esses dados de entrada em um conjunto de informações significativas para determinado propósito.

Primeiro vamos ver os tipos de campos para montar um formulário, e depois passaremos aos *scripts*.

SELECT

<SELECT> apresenta uma lista de valores, através de campos `OPTION`.

```
<SELECT                                                    NAME="sabor">
<OPTION>Abacaxi
<OPTION>Creme
<OPTION>Morango
<OPTION>Chocolate
</SELECT>
```

Também é possível estabelecer uma escolha-padrão, através do atributo `SELECTED`

```
<SELECT                                                    NAME="sabor">
<OPTION>Abacaxi
<OPTION                                                  SELECTED>Creme
<OPTION>Morango
<OPTION>Chocolate
</SELECT>
```

Com o atributo `SIZE`, escolhe-se quantos itens da lista serão mostrados (no exemplo, **SIZE**=4):

```
<SELECT                              NAME="sabor"                          SIZE=4>
<OPTION>Abacaxi
<OPTION                                                  SELECTED>Creme
<OPTION>Morango
<OPTION>Chocolate
</SELECT>
```

Com o atributo `MULTIPLE`, define-se que se pode selecionar mais de um item (pressionando a tecla Shift do teclado enquanto se selecionam os itens):

```
<SELECT                NAME="sabor"              SIZE=4                MULTIPLE>
<OPTION>Abacaxi
<OPTION                                                  SELECTED>Creme
<OPTION>Morango
<OPTION>Chocolate
</SELECT>
```

TEXTAREA

<TEXTAREA> abre uma área para entrada de texto, de acordo com atributos para número de colunas, linhas, e - se for o caso - um valor inicial.

```
<TEXTAREA COLS=40 ROWS=5 NAME="comentario"> Deixe seu comentário </TEXTAREA>
```

CGI Scripts

`CGI`, ou *Common Gateway Interface*, é uma interface definida de maneira a possibilitar a execução de programas - "gateways" - sob um servidor HTTP. Neste contexto, os "gateways" são programas ou scripts (também chamados "cgi-bin") que recebem requisições de informação, retornando um documento com os resultados correspondentes. Esse documento retornado pode existir previamente, ou pode ser gerado pelo script especialmente para atender àquela requisição específica do usuário (diz-se que o documento é ´gerado *"on the fly"*´).
Exemplos de aplicação de CGI incluem:

- processamento de dados submetidos através de formulários: consulta a banco de dados, cadastramento em listas, livros de visita, pesquisas de opinião;
- criação de documentos personalizados *on the fly*;
- gerenciamento de contadores de acesso;
- processamento de mapas.

Tais scripts podem ser escritos em qualquer linguagem que possa ler variáveis, processar dados e retornar respostas - ou seja, qualquer linguagem de programação! A linguagem é determinada de acordo com a plataforma do servidor e da aplicação a ser implementada.

Visão Geral

Os scripts têm uma forma geral comum:

1. leitura de dados entrados pelo usuário (e/ou campos de informação de um pacote HTTP);
2. processamento dos dados (armazenamento dos dados em um banco de dados, realização de cálculos, recuperação de dados etc.);
3. montagem de uma página Web ou geração de uma imagem com os resultados produzidos.

Submissão de formulários

Suponhamos um formulário cuja marcação principal seja:

```
<FORM ACTION="/cgi-bin/teste.cgi" METHOD=método>
...
</FORM>
```

onde `ACTION` indica o `URL` do script que receberá os dados (ainda não vamos nos preocupar com o campo `METHOD`)

Quando submetemos os dados de um formuláriom, o browser organiza os dados entrados pelo usuário, assegurando que as informações chegarão em ordem e serão compreendidas pelo script. Vejamos alguns exemplos.

Um campo de texto simples, tal como:

```
<INPUT TYPE=TEXT NAME="nome">
```

cuja entrada tenha sido...

será organizado pelo browser da seguinte forma:

```
nome=Prof.+Achille+Bassi
```

Isto é, passa a assumir um formato `nome = valor`, onde:

> **nome** foi definido nas etiquetas do formulário (pelo atributo `NAME` de cada campo) e **valor** é a entrada oferecida pelo usuário.

Note que os espaços em branco são substituídos por sinais de +. Se o formulário tiver mais algum campo de informação, por exemplo:

```
<INPUT TYPE=TEXT NAME="email">
```

com uma entrada

o browser estará gerando uma linha única com todos esses dados, desta forma:

```
nome=Prof.+Achille+Bassi&email=bassi@icmc.usp.br
```

Isto é, os campos de informação são separados por `&`. Também todos os acentos e símbolos especiais são codificados em hexadecimal para o envio dos dados. Esta codificação dos dados de um formulário é padrão.

Feita essa codificação, o browser envia uma requisição para o `URL` indicado em `ACTION`, que está associado a um script. Esse envio está sujeito a um método específico, que definirá como o script irá recuperar essas informações para processá-las.

Antes de ver os métodos, é bom conhecer as variáveis de ambiente, que serão muito úteis no tratamento de informações.

CGI Scripts usando PHP

A principal diferença entre o uso de Perl e PHP para criar páginas dinâmicas é que o PHP pode ser totalmente embutido em uma página comum de HTML. Assim, não é preciso fazer com que o script gere uma página de resposta com as formatações: os comandos de PHP podem ficar embutidos de maneira bem elegante e intuitiva entre as marcações de HTML.

Abaixo de cada campo do formulário foi colocado um exemplo da formatação usada para criá-lo. Ao fim da página está a formatação completa do formulário.

Se você tiver dúvidas sobre essas formatações, veja antes a seção sobre formulários.

```
<form action=form1.php method=post>
```

`form1.php` será o script em PHP que irá trabalhar com os dados entrados por meio deste formulário; ele irá escrever, na próxima página, as informações entradas neste formulário.

Nome:	`<input type="text" name="nome" size="35">`
E-mail:	`<input type="text" name="email" size="25">`
Esportes que você pratica:	Futebol Vôlei Natação Basquete Tênis Bocha `<input type="checkbox" name="esporte1" value="futebol">Futebol ` `<input type="checkbox" name="esporte2" value="volei" checked>Vôlei `

Seu time do coração:	Palmeiras; Náutico; Flamengo; Grêmio; Santos; Atlético; Corinthians; Fluminense; Internacional; Cruzeiro; Botafogo; Santa Cruz; São Paulo; AEA.
	`<input type=radio name="time" value="palmeiras">Palmeiras;`
	`<input type=radio name="time" value="nautico">Náutico;`
	`<input type=radio name="time" value="fla">Flamengo;`

Sabor de sorvete que prefere:	`<select value=sabor name="sorvete">`
	`<option>Abacaxi`
	`<option>Creme`

```
<option>Morango
<option>Chocolate
</select>
```

Algo mais?

```
<textarea cols=40 rows=5 name="comentario">Escreva um comentário</textarea>

<input type="submit" name="submit" value="Enviar">
<input type="reset" name="cancela" value="Apagar">
```

Note que, na entrada de dados sobre esportes, cada item tem um nome diferente (esporte1, esporte2,...), pois mais de um esporte pode ser escolhido.
No caso da entrada sobre o time preferido, todos os itens têm o mesmo nome (time), pois somente um poderá ser selecionado.

Experimente entrar dados nesse formulário - ou apenas pressione "Enviar" - e veja como é conseguido o resultado.

Aqui está a formatação completa do formulário acima, abrangendo uma tabela:

```
<form name="form1" action=form1.php method=post>
<table class="conteudo" border="0" cellpadding="5" cellspacing="0" width="95%"
align="center">
<tr>
<td valign="top" bgcolor="#CCCCCC">
<p class="conteudo">Nome:</p></td>
<td bgcolor="#CCCCCC">
<p class="conteudo"><input type="text" name="nome" size="35"></p>
</td>
</tr>
<tr>
<td valign="top">
<p>E-mail:</p></td>
<td><p class="conteudo"><input type="text" name="email" size="25"></p>
</td>
</tr>
<tr>
<td valign="top" bgcolor="#CCCCCC">
<p class="conteudo">Esportes que você pratica:</p>
</td>
<td bgcolor="#CCCCCC">
<p class="conteudo"><input type="checkbox" name="esporte1"
value="futebol">Futebol<br><input type="checkbox" name="esporte2" value="volei"
checked>Vôlei<br><input type="checkbox" name="esporte3" value="natacao"
checked>Natação<br><input type="checkbox" name="esporte4"
value="basquete">Basquete<br><input type="checkbox" name="esporte5"
value="tenis">Tênis<br><input type="checkbox" name="esporte6" value="bocha">Bocha</p>
</td>
</tr>
<tr>
<td valign="top">
<p class="conteudo">Seu time do coração:</p>
</td>
<td>
<p class="conteudo"><input type=radio name="time" value="palmeiras">Palmeiras; <input
type=radio name="time" value="nautico">Náutico; <input type=radio name="time"
value="fla">Flamengo; <input type=radio name="time" value="gremio">Grêmio; <input
type=radio name="time" value="santos">Santos; <input type=radio name="time"
value="atletico">Atlético; <input type=radio name="time"
value="corinthians">Corinthians; <input type=radio name="time"
value="flu">Fluminense; <input type=radio name="time" value="inter">Internacional;
<input type=radio name="time" value="cruzeiro">Cruzeiro; <input type=radio
```

```
name="time" value="botafogo">Botafogo; <input type=radio name="time"
value="santa">Santa Cruz; <input type=radio name="time" value="saopaulo">São Paulo;
<input type=radio name="time" value="aea" checked>AEA.</p>
</td>
</tr>
<tr>
<td valign="top" bgcolor="#CCCCCC">
<p class="conteudo">Sabor de sorvete que prefere:</p></td>
<td bgcolor="#CCCCCC"><select value=sabor name="sorvete" size="1">
<option>Abacaxi
<option>Creme
<option>Morango
<option>Chocolate
</select></td>
</tr>
<tr>
<td valign="top"><p class="conteudo">Algo mais? </p></td>
<td><textarea cols=40 rows=5 name="comentario">Escreva um comentário</textarea></td>
</tr>
<tr>
<td valign="top" bgcolor="#CCCCCC">
<p> </p>
</td>
<td bgcolor="#CCCCCC"><p class="conteudo"> <input type="submit" name="submit"
value="Enviar"> <input type="reset" name="cancela" value="Apagar"></p>
</td>
</tr>
</table>
</form>
```

Áudio e Vídeo

O uso de áudio e vídeo na Internet vem ganhando muito destaque nos últimos dois anos, e é bom saber como usar bem estas mídias.

Áudio e vídeo são classificados como "mídias contínuas", pois são geradas segundo determinadas taxas, devendo ser reproduzidas nessa mesma taxa, para evitar distorções. Quanto mais informações de uma seqüência de áudio ou vídeo digital são armazenados, melhor a qualidade do áudio ou vídeo reproduzido. Isso implica, logicamente, no fato de arquivos de áudio e vídeo serem geralmente muito grandes, o que torna inviável o uso mais freqüente dessas mídias em páginas Web.

Além de procurarmos lidar sempre com pequenos trechos de áudio e vídeo, podemos explorar tecnologias recentes que permitem a transmissão em tempo real.

Folhas de Estilo

Um avanço interessante na linguagem HTML após a versão 3.2 foi a introdução das *Style Sheets* ou *Cascading Style Sheets*. Essas *folhas de estilo* permitem o uso de formatações uniformes em um site, de maneira bastante "econômica".
Logo nas primeiras seções deste tutorial (Cores e fontes de textos), vimos que, para poder formatar um texto, era preciso escrever uma marcação parecida com:

```
<h3><font face="Arial" color="#4A7D7B">Um título genérico</font></h3>
<p><font face="Arial" size="2">Um texto genérico com algum </font><font
face="Arial" size="2" color="red">destaque qualquer</font><font face="Arial"
size="2"> junto, após um título genérico, etc.</font></p>
```

para ter o resultado:

Um título genérico

Um texto genérico com algum destaque qualquer junto, após um título genérico, etc.

Acontece que, de um documento para outro, pode acontecer de esquecermos o tamanho da fonte usada no texto, qual a fonte ou a cor dos títulos de determinada subseção, e a uniformidade de apresentação das páginas acaba ficando prejudicada.
Com as folhas de estilo, podemos declarar estilos apropriados para seções de texto, aplicando esses estilos sem nos preocuparmos com detalhes de fontes, cor e tamanhos.
E mais: se for necessário modificar algum dia as cores de todos os títulos ou dos destaques ao longo dos textos, simplesmente alteramos a folha de estilo, atualizando imediatamente a apresentação de todos os documentos que fazem referência a ela.
Para o exemplo acima, poderíamos criar a seguinte folha de estilo:

```
BODY            {            font:            10pt            Arial            }
H3                    {                    color:#4A7D7B                    }
.destaque { color: red }
```

E assim, para ter o mesmo resultado do exemplo acima, a formatação seria muito mais simples que a primeira:

```
<h3>Um                          título                          genérico</h3>
<p>Um texto genérico com algum <span class="destaque">destaque qualquer</span>
junto, após um título genérico, etc.</p>
```

A definição da folha de estilo deve ficar dentro de <HEAD>, da seguinte forma, se a folha for definida dentro do mesmo documento em que está sendo usada:

```
<HEAD>
...
<STYLE                                                    TYPE="text/css">
   BODY           {           font:           10pt           Arial           }
   H3                   {                   color:#4A7D7B                   }
   .destaque             {             color:             red             }
</STYLE>
...
</HEAD>
```

Ou então,quando a folha de estilo é definida externamente:

```
<HEAD>
...
<LINK               REL="stylesheet"               HREF="folha_de_estilo.css">
...
</HEAD>
```

Neste caso, uma mesma folha de estilo pode ser usada por vários documentos HTML, que também poderão ter suas próprias folhas de estilo internas.

Introdução as CSS

O HTML atual

E, já com várias inovações o HTML era usado para construção de páginas Web, que no início limitavam-se a exibir informações contidas nos documentos.

A evolução vinha atropelando tudo com uma avalanche de novos aplicativos, facilidades, softwares, hardware etc. E o HTML não passou ao largo, pelo contrário, a simples linguagem de marcação destinada a apresentar conteúdos carecia de uma maior flexibilidade no sentido de manipular visualmente os conteúdos.

Novas tags e atributos foram inventados, tais como a tag `font` e o atributo `color` que permitiam alterar a aparência de textos.
Assim nasceu a **estilização** dos conteúdos.

E a evolução trazendo novas descobertas, corre célere neste dinamismo alucinante que estamos testemunhando até os dias de hoje. Novas tags e novos atributos de estilo foram introduzidos no HTML. Com isso, a velha linguagem de marcação passou a exercer uma dupla função. A função de estruturar o conteúdo através da marcação e a função de apresentá-lo ou seja de dar a aparência final.

Os problemas criados

Mas, esta dupla função do HTML, se por um lado resolveu uma necessidade dos designers e projetistas por outro acabou trazendo sérios problemas aos projetos criados. Os documentos Web publicados na Internet, cada vez mais sofisticados e extensos, estavam fugindo do controle de seus criadores.

Para ilustrar suponha o seguinte exemplo:

Seu melhor cliente telefona às 17:00h da tarde de uma sexta-feira (sempre ligam nesta hora para solicitar alguma coisa não é mesmo?) e diz o seguinte;

teremos uma reunião aqui na empresa, na segunda-feira às 0800h com um potencial comprador e é nossa intenção fazer uma apresentação dos nossos produtos através do site que você criou e mantém. Seguindo uma sugestão do nosso departamento de marketing precisamos mudar a cor de todos os títulos no site de verde para vermelho, pois que esta é a cor principal da marca do nosso comprador e com isso pretendemos fixar uma cumplicidade subliminar. Isto é bem simples de fazer, não é? Afinal é só mudar a cor! Dá para você 'botar no ar' até às 19:30h ? Quero dar uma olhadinha antes de encerrar o expediente. OK?

Claro que você concorda e responde que vai providenciar rapidinho, afinal *é só para mudar a cor*. Mas, são 180 páginas no site! E os títulos são tags de cabeçalho deste tipo:

```
<h1><font color="#00FF00">Título</font></h1>
```

Supondo uma média de 3 títulos por página, você tem um total de 540 tags `font` para editar e mudar o atributo `color`. E se o seu cliente tivesse pedido para mudar a cor dos textos, e do fundo? Bem, este exemplo simples dá uma dimensão de um dos problemas criados com a mistura de marcação com apresentação - estilização!

A solução proposta

Cada vez mais ficava evidente que esta mistura que maravilhou os projetistas Web no início, tornara-se uma grande dor de cabeça. E é claro, a solução passava por dissociar linguagem de marcação da estilização.

Desta necessidade, eu diria mesmo uma imposição, nasceu as CSS, sigla em inglês para *Cascading Style Sheet* que em português foi traduzido para Folha de Estilo em Cascata.

A introdução deste conceito preconiza o uso dos elementos (tags) HTML, exclusivamente para marcar e estruturar o conteúdo do documento. Nenhum elemento HTML será usado para alterar a apresentação, ou seja estilizar o conteúdo.
A tarefa de estilização ficará a cargo das CSS que nada mais é do que um arquivo independente do arquivo HTML no qual são declaradas propriedades e valores de estilização para os elementos do HTML.

Estas declarações de estilo, quer sejam estruturadas em um arquivo externo com extensão `.css` quer sejam declaradas de outros modos (importadas, lincadas, incorporadas ou inline), contém todas as regras de estilo para os elementos do documento HTML.

Voltando àquela situação criada no item anterior, agora você mudaria a cor de TODOS os cabeçalhos `h1` em TODO o site em CINCO SEGUNDOS. Às 19:20h você retorna a ligação do cliente e pede para a secretária avisá-lo de que "já está no ar", sem maiores traumas, correrias e estresses. Ah e mais, mesmo que o site tivesse 1.800 páginas e não as 180 da situação criada, você gastaria os mesmos cinco segundos.

As restrições

A idéia, a filosofia mesmo, de projeto Web aponta para uso amplo das CSS, ainda não explorada em toda sua potencialidade por razões de incompatibilidades de certas propriedades CSS com navegadores mais antigos e com as interpretações diferentes das CSS por parte das aplicações de usuários criadas por fabricantes distintos.

Contudo, há uma tendência - e torcemos para que se concretize rapidamente - de que as novas tecnologias voltadas para o desenvolvimento, não só das variadas aplicações de usuário como também de softwares e hardwares, atendam e se enquadrem dentro das recomendações e especificações dos órgãos normatizadores, notadamente as standards do W3C.

Quando o projeto Web em todas as suas incontáveis variantes, seguir a normatização e padronização recomendada pelo W3C, teremos uma Web muito mais fácil, dinâmica e agradável.

O efeito cascata

Que estilo será aplicado, quando há conflito de estilos especificados (por exemplo: uma regra de estilo determina que os parágrafos serão na cor preta e outra que serão na cor azul) para um mesmo elemento HTML?

Aqui entra o **efeito cascata,** que nada mais é, do que o estabelecimento de uma *prioridade* para aplicação da regra de estilo ao elemento.
Para determinar a prioridade são considerados diversos fatores, entre eles, o tipo de folha de estilo, o local físico da folha de estilo no seu todo, o local físico da regra de estilo na folha de estilo e a especificidade da regra de estilo.

A prioridade para o efeito cascata em ordem crescente:

1. folha de estilo padrão do navegador do usuário;
2. folha de estilo do usuário;
3. folha de estilo do desenvolvedor;
    - estilo externo (importado ou linkado).
    - estilo incorporado (definido na seção `head` do documento);
    - estilo inline (dentro de um elemento HTML);
4. declarações do desenvolvedor com !important;
5. declarações do usuário com !important;

Assim, uma declaração de estilo com !important definido pelo usuário prevalece sobre todas as demais, é a de mais alta prioridade. Entre as folhas de estilo definidas pelo desenvolvedor do site, os estilos inline (dentro de um elemento HTML) tem a prioridade mais elevada, isto é, prevalecerá sobre a folha de estilo definida na seção `head`, e, esta prevalecerá sobre uma folha de estilo externa. A prioridade mais baixa é para estilos padrão do navegador.

Agora você já sabe o porquê de "cascata" no nome Folha de estilo em cascata.
Consulte os diversos tutoriais deste site para saber mais sobre o efeito cascata.

A regra CSS

A regra CSS e sua sintaxe

Uma regra CSS é uma declaração que segue uma sintaxe própria e que define como será aplicado estilo a

um ou mais elementos HTML . Um conjunto de regras CSS formam uma Folha de Estilos. Uma regra CSS, na sua forma mais elementar, compõe-se de três partes: um seletor, uma propriedade e um valor e tem a sintaxe conforme mostrado abaixo:

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

Seletor: genericamente, é o elemento HTML identificado por sua tag, ou por uma classe, ou por uma ID, ou etc., e para o qual a regra será válida (por exemplo: `<p>`, `<h1>`, `<form>`, `.minhaclasse`, etc...);

Propriedade: é o atributo do elemento HTML ao qual será aplicada a regra (por exemplo: font, color, background, etc...).

Valor: é a característica específica a ser assumida pela propriedade (por exemplo: letra tipo arial, cor azul, fundo verde, etc...)

Na sintaxe de uma regra CSS, escreve-se o seletor e a seguir a propriedade e valor separados por dois pontos e entre chaves { }. Quando mais de uma propriedade for definida na regra, deve-se usar ponto-e-vírgula para separá-las. O ponto-e-vírgula é facultativo no caso de propriedade única e também após a declaração da última propriedade no caso de mais de uma.

No entanto é de boa técnica usar-se sempre o ponto-e-vírgula após cada regra para uma propriedade.

Ver os exemplos abaixo:

```
p {
font-size: 12px; /* ponto-e-vírgula é facultativo */
}

body {
color: #000000;
background: #FFFFFF;
font-weight: bold; /*ponto-e-vírgula é facultativo */
}
```

No exemplo abaixo, o **seletor** é o "documento todo" (body - a página web), a **propriedade** é o fundo do documento e o **valor** é a cor branca.

```
body {
background: #FFFFFF;
}
```

Se o valor for uma palavra composta, deverá estar entre aspas duplas " ", ou simples ':

```
h3 {
font-family: "Comic Sans MS"
}
```

Para **maior legibilidade** das folhas de estilo, é de boa prática usar linhas distintas para escrever cada uma das declarações — propriedade e seu valor —, como mostrado abaixo:

```
p {
text-align: right;
color: #FF0000;
}
```

Isto não é obrigatório! A regra abaixo tem o mesmo efeito da regra acima e ambas as sintaxes estão corretas:

```
p {text-align: right;color: #FF0000;}
```

NOTA: A razão do uso de ponto e vírgula na última declaração ou mesmo quando só há uma declaração é que durante a fase de projeto da Folha CSS quase sempre estaremos acresentando novas declarações e a última declaração quase nunca é a última na fase de projeto. Assim, esta prática certamente nos poupará revisões por ter esquecido um ponto e vírgula.!!!!

Agrupamento de Seletores

Uma regra CSS quando válida para vários seletores, estes podem ser agrupados. Separe cada seletor com uma vírgula. No exemplo abaixo agrupamos todos os elementos cabeçalho. A cor de todos os cabeçalhos será verde.

```
h1, h2, h3, h4, h5, h6 {
color: #00FF00;
}
```

O seletor classe

Mas você não está restrito somente aos elementos HTML (tags) para aplicar regras de estilo!

Você pode "inventar" um nome e com ele criar uma **classe** a qual definirá as regras CSS. E o mais interessante das classes, é que elas podem ser aplicadas a **qualquer elemento** HTML. E mais ainda, você pode aplicar estilos diferentes para o mesmo tipo de elemento do HTML, usando classes diferentes para cada um deles.

A sintaxe para o seletor classe é mostrada abaixo. Elemento HTML mais um nome qualquer que você "inventa" precedido de . (ponto):

```
 elemento HTML.minhaclasse {
propriedade: valor;
}
```

Nota: Para o nome que você "inventa" evite usar números e caracteres especiais. Tanto quanto possível use só letras de a-z e de A-Z. Há restrições quanto ao uso de números e caracteres. Minha experiência e conselho: Use só letras!

Por exemplo: suponha que você queira ter dois tipos de parágrafos em seu documento: um parágrafo com

letras na cor preta e um parágrafo com letras na cor azul.

```
p.corum {
color:#000000;
}

p.cordois {
color:#0000FF;
}
```

No seu documento HTML as regras seriam aplicadas conforme abaixo:

```
<p class ="corum"> este parágrafo terá cor preta.</p>

<p class ="cordois">
este parágrafo terá cor azul.
</p>
```

Este item foi atualizado em 2007/07/03

Em CSS 1 não é válido atribuir mais de uma classe para um elemento HTML. O exemplo abaixo está errado:

```
<p class ="corum" class ="cordois">
Aqui há um erro.
</p>
```

Nota: CSS 2 mudou este conceito, permitindo declarar mais de uma classe, desde que o nome das classes sejam separados por um espaço.

```
<p class ="corum cordois">
Aqui não há erro.
</p>
```

Ao criar uma classe você talvez queira que ela seja aplicável a qualquer elemento HTML. Neste caso basta que se omita o nome do elemento antes da classe. Por exemplo: a regra CSS a seguir pode ser aplicada a qualquer elemento HTML ao qual você deseja atribuir cor azul:

```
.cortres {
    color: #0000FF;
    }
```

No exemplo a seguir tanto o cabeçalho `<h2>` como o parágrafo `<p>` terão cor azul:

```
<h2 class="cortres">
Este cabeçalho é azul.
</h2>

<p class="cortres">
Este parágrafo é azul.
</p >
```

O seletor ID

O seletor ID difere do seletor de classe, por ser ÚNICO. Um seletor ID só pode ser aplicado a UM e somente UM elemento HTML dentro do documento.

Você pode "inventar" um nome e com ele criar uma **ID** a qual definirá as regras CSS. Uma **ID só pode ser aplicada a UM elemento** HTML.

A sintaxe para o seletor ID é mostrada abaixo. Um nome qualquer que você "inventa" precedido de # ("tralha", "jogo-da-velha" :-)):

```
#minhaID {
propriedade: valor;
        }
```

Nota: Para o nome que você "inventa" evite usar números e caracteres especiais. Tanto quanto possível use só letras de a-z e de A-Z. Há restrições quanto ao uso de números e caracteres. Minha experiência e conselho: Use só letras!

Inserindo comentários nas CSS

Você pode inserir comentários nas CSS para explicar seu código, e principalmente ajudá-lo a relembrar de como você estruturou e qual a finalidade de partes importantes do código. Daqui há alguns meses a menos que você seja um privilegiado, terá esquecido a maior parte daquilo que você levou horas para "bolar". O comentário introduzido no código, será ignorado pelo browser. Um comentário nas CSS começa com o "/*", e termina com " */". Veja o exemplo abaixo:

```
/* este é um comentário*/
p {
font-size: 14px;       /* este é outro comentário*/
color: #000000;
font-family: Arial, Serif;
}
```

Vinculando folhas de estilo a documentos

Os três tipos de vinculação de folhas de estilo

As folhas de estilo podem ser vinculadas a um documento de três maneiras distintas:

1. Importadas ou lincadas;
2. Incorporadas;
3. Inline.

Folha de estilo externa

Uma folha de estilo é dita externa, quando as regras CSS estão declaradas em um documento a parte do documento HTML. A folha de estilo é um arquivo separado do arquivo html e que tem a extensão .css

Uma folha de estilo externa é ideal para ser aplicada a várias páginas. Com uma folha de estilo externa , você pode mudar a aparência de um site inteiro mudando um arquivo apenas (o arquivo da folha de estilo).

O arquivo css da folha de estilo externa deverá ser lincado ou importado ao documento HTML, dentro da tag <head> do documento. A sintaxe geral para lincar uma folha de estilo chamada estilo.css é mostrada abaixo.

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

A sintaxe geral para importar uma folha de estilo chamada estilo.css é mostrada abaixo:

```
<head>
...........
<style type="text/css">
@import url("estilo.css");
</style>
..........
</head>
```

O browser lerá as regras de estilo do arquivo estilo.css, e formatará o documento de acordo com elas.

Uma folha de estilo externa pode ser escrita em qualquer editor de texto. O arquivo não deve conter nenhuma tag HTML. As folhas de estilo devem ser "salvas" com uma extensão .css

Folha de estilo incorporada ou interna

Uma folha de estilo é dita incorporada ou interna, quando as regras CSS estão declaradas no próprio documento HTML.

Uma folha de estilo incorporada ou interna, é ideal para ser aplicada a uma única página. Com uma folha de estilo incorporada ou interna, você pode mudar a aparência de somente um documento, aquele onde a folha de estilo esta incorporada.

As regras de estilo, válidas para o documento, são declaradas na seção <head> do documento com a tag de estilo <style>, conforme sintaxe mostrada abaixo:

```
<head>
...........
<style type="text/css">
<!--
body {
background: #000000;
url("imagens/minhaimagem.gif");
}
h3 {
color: #FF0000;
}
p {
margin-left: 15px;
padding:1.5em;
}
-->
</style>
...........
</head>
```

O browser lerá agora as regras de estilo na própria página, e formatará o documento de acordo com elas.

Nota: Um browser ignora normalmente as tags desconhecidas. Isto significa que um browser velho que não suporte estilos, ignorará a tag <style>, mas o conteúdo da tag será mostrado na tela. É possível impedir que um browser velho mostre o conteúdo da tag, "escondendo-o" através do uso da marcação de comentário do HTML.

Observe a inclusão dos símbolos <!-- (abre comentário) --> (fecha comentário) no código acima.

Folha de estilo inline

Uma folha de estilo é dita inline, quando as regras CSS estão declaradas dentro da tag do elemento HTML.

Um estilo inline só se aplica a um elemento HTML. Ele perde muitas das vantagens de folhas de estilo pois mistura o conteúdo com a apresentação. Use este método excepcionalmente, como quando quiser aplicar um estilo a uma única ocorrência de um elemento.

A sintaxe para aplicar estilo inline é mostrada a seguir:

```
<p style="color:#000000; margin: 5px;">
Aqui um parágrafo de cor preta e com 5px nas 4  margens.
</p>
```

Folhas múltiplas de estilo

Se alguma propriedade for definida para um mesmo elemento em folhas de estilo diferentes, entrará em ação, o EFEITO CASCATA e prevalecerão os valores da folha de estilo mais específica.

Suponha, uma folha de estilo externa com as seguintes propriedades para o seletor h2:

```
h2 {
color: #FFCC00;
text-align: center;
font: italic 9pt Verdana, Sans-serif;
}
```

e, uma folha de estilo interna com as seguintes propriedades para o seletor h2:

```
h2 {
color: #FFCC00;
text-align: center;
font: italic 10pt Verdana, Sans-serif;
}
```

Se ambas as páginas estiverem vinculadas ao documento, como há um conflito no tamanho das letras para <h2>, prevalecerá a folha interna e a letra de <h2> terá o tamanho igual a 10 pt.

A propriedade font

As fontes nos elementos HTML

As propriedades para as fontes, definem as características (os valores na regra CSS) das letras que constituem os textos dentro dos elementos HTML.

As propriedades básicas para fontes e que abordaremos neste tutorial são as listadas abaixo:

- color:.....cor da fonte
- font-family:.....tipo de fonte
- font-size:.....tamanho de fonte

- font-style:.....estilo de fonte
- font-variant:.....fontes maiúsculas de menor altura
- font-weight:.....quanto mais escura a fonte é (negrito)
- font:.....maneira abreviada para todas as propriedades

Valores válidos para as propriedades da fonte

- **color:**
    1. código hexadecimal: #FFFFFF
    2. código rgb: rgb(255,235,0)
    3. nome da cor: red, blue, green...etc
- **font-family:**
    1. nome da família de fontes : define-se pelo nome da fonte, p. ex:"verdana", "helvetica", "arial", etc.
    2. nome da família genérica: define-se pelo nome genérico, p. ex:"serif", "sans-serif", "cursive", etc.
- **font-size:**
    1. xx-small
    2. x-small
    3. small
    4. medium
    5. large
    6. x-large
    7. xx-large
    8. smaller
    9. larger
    10. length: uma medida reconhecida pelas CSS (px, pt, em, cm, ...)
    11. %
- **font-style:**
    1. normal: fonte normal na vertical
    2. italic: fonte inclinada
    3. oblique:fonte obliqua
- **font-variant:**

1. normal: fonte normal
2. small-caps: transforma em maiúsculas de menor altura

- **font-weight:**
    1. normal
    2. bold
    3. bolder
    4. lighter
    5. 100
    6. 200
    7. 300
    8. 400
    9. 500
    10. 600
    11. 700
    12. 800
    13. 900

Vamos a seguir analisar cada uma delas detalhadamente através de exemplos práticos.

Como estudar e entender os exemplos

Para cada propriedade apresento as regras CSS para um ou mais elementos HTML e definidas dentro de uma folha de estilos incorporada, bem como um trecho do documento HTML onde se aplicam as regras.

A seguir mostro o efeito que a regra produz. Observe a regra e o efeito e para melhor fixar seu aprendizado reproduza o código no seu editor, mude os valores e veja o resultado no browser. Esta é a melhor e mais rápida maneira de você aprender CSS. Bons estudos! E faça ótimo proveito dos tutoriais.

color ... A cor da fonte

```
<html>
<head>
<style type="text/css">
<!--
h1 {color: #FF0000;}
h2 {color: #00FF00;}
p {color: rgb(0,0,255);}
-->
</style>
</head>
<body>
```

```
<h1>Cabeçalho com letras vermelhas</h1>
<h2>Cabeçalho com letras verdes</h2>
<p>Parágrafo com letras azuis</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Cabeçalho com letras vermelhas

Cabeçalho com letras verdes

Parágrafo com letras azuis

font-family...O tipo de fonte

```
<html>
<head>
<style type="text/css">
<!--
h2 {font-family: arial, helvetica, serif;}
p {font-family: sans-serif;}
-->
</style>
</head>
<body>
<h2>Família por nome: arial, helvetica, serif;</h2>
<p>Família genérica: sans-serif;<p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Família por nome: arial, helvetica, serif;

Família genérica: sans-serif;

Notas: A propriedade font-family é usada para definir uma lista de tipos de fontes.

O browser assume o primeiro nome que ele reconhece na lista.

Separar cada nome por uma vírgula e sempre prever um nome genérico.

Se o nome da fonte for composto (mais de uma palavra, p. ex: Comic Sans MS), usar aspas duplas no

nome. Se estiver definindo um atributo de "style" em HTML, onde as aspas duplas já estão presentes usar no nome de fonte composto, aspas simples.

font-size...O tamanho de fonte

```
<html>
<head>
<style type="text/css">
<!--
h1 {font-size: 14px;}
h2 {font-size: smaller;}
p {font-size: 100%;}
-->
</style>
</head>
<body>
<h1>Letras com tamanho: 14px</h1>
<h2>Letras com tamanho: smaller</h2>
<p>Letras com tamanho:100%</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Letras com tamanho: 14px

Letras com tamanho: smaller

Letras com tamanho:100%

font-style...O estilo de fonte

```
<html>
<head>
<style type="text/css">
<!--
h1 {font-style: italic;}
h2 {font-style: normal;}
p {font-style: oblique;}
-->
</style>
</head>
<body>
<h1>Este é o estilo italic</h1>
<h2>Este é o estilo normal</h2>
<p>Este é o estilo oblique</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Este é o estilo italic

Este é o estilo normal

Este é o estilo oblique

font-variant...fontes maiúsculas "menores"

```
<html>
<head>
<style type="text/css">
<!--
h3 {font-variant: normal;}
p{font-variant: small-caps;}
-->
p</style>
</head>
<body>
<h3>Este cabeçalho com letras normais</h3>
<p>Este parágrafo com letras em "small-caps"</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Este cabeçalho com letras normais

ESTE PARÁGRAFO COM LETRAS EM "SMALL-CAPS"

font-weight...Peso das fontes - negrito (intensidade da cor)

```
<html>
<head>
<style type="text/css">
<!--
p {font-weight: bold;}
-->
</style>
</head>
<body>
<p>
Este é um parágrafo em negrito</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Este é um parágrafo em negrito

font...Todas as propriedades das fontes em uma declaração única

Esta é a maneira abreviada de você escrever uma regra para as propriedades das fontes.

A sintaxe geral é esta: `font: [style] [variant] [weight] [`**size**`] [/line-height] [`**family**`] | caption | icon | menu | message-box | small-caption | status-bar | inherit`

Você pode declarar todas ou algumas das propriedades.

Os valores `size` e `family` são obrigatórios. Os demais são facultativos (se você os omitir será adotado o valor default ou herdado do elemento pai).

Os valores `style, variant, weight` e `size`, podem ser declarados em qualquer ordem.

```
<html>
<head>
<style type="text/css">
<!--
p {
font: italic small-caps bold 14px
 "Comic Sans MS", sans-serif;
}
-->
</style>
</head>
<body>
<p>Parágrafo em declaração única</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

PARÁGRAFO EM DECLARAÇÃO ÚNICA

O valores `caption, icon, menu, message-box, small-caption` e `status-bar` referem-se às fontes usadas pelo sistema operacional do visitante do site e devem ser declarados **isolados** na propriedade `font`.

- caption.....fontes usadas em botões;
- icon.....fontes usadas em ícones;
- menu.....fontes usadas em menus;
- message-box...fontes ussadas em caixas de mensagens;
- small-caption...fontes usadas em pequenos controles;
- status-bar.....fontes usadas na barra de status;

O valor `inherit` é usado para herdar a fonte usada pelo elemento pai e também deve ser declarados **isolados** na propriedade `font`.

```
<html>
<head>
<style type="text/css">
<!--
.p1 {
font: status-bar;
}
.p2 {
font: inherit;
.p3 {
font: small-caption ;
}
}
-->
</style>
</head>
<body>
<p class="p1">Parágrafo com fonte status-bar</p>
<p class="p2">Parágrafo com fonte inherit</p>
<p class="p3">Parágrafo com fonte small-caption</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Parágrafo com fonte status-bar

Parágrafo com fonte inherit

Parágrafo com fonte small-caption

Você poder fazer uso de um excelente editor para a propriedade font e descobrir mais efeitos para complementar este tutorial, nesta página interativa.

A propriedade text

Os textos nos elementos HTML

As propriedades para textos, definem as características (os valores na regra CSS) dos textos inseridos dentro dos elementos HTML.

As propriedades para textos são as listadas abaixo:

- color.....cor do texto;
- letter-spacing.....espaçamento entre letras;

- word-spacing.....espaçamento entre palavras;
- text-align.....alinhamento do texto;
- text-decoration.....decoração do texto;
- text-indent.....recuo do texto;
- text-transform.....forma das letras;
- direction.....direção do texto;
- white-space.....como o browser trata os espaços em branco;

Valores válidos para as propriedades do texto

- **color:**
    1. código hexadecimal: #FFFFFF
    2. código rgb: rgb(255,235,0)
    3. nome da cor: red, blue, green...etc
- **letter-spacing:**
    1. normal: é o espaçamento default
    2. lenght: uma medida reconhecida pelas CSS (px, pt, em, cm, ...) São válidos valores negativos
- **word-spacing:**
    1. normal: é o espaçamento default
    2. lenght: uma medida reconhecida pelas CSS (px, pt, em, cm, ...) São válidos valores negativos
- **text-align:**
    1. left: alinha o texto a esquerda
    2. right: alinha o texto a direita
    3. center: alinha o texto no centro
    4. justify: força o texto a ocupar toda a extensão da linha da esquerda a direita
- **text-decoration:**
    1. none: nenhuma decoração
    2. underline: coloca sublinhado no texto
    3. overline: coloca um sobrelinhado no texto
    4. line-through: coloca uma linha em cima do texto
    5. blink: faz o texto piscar

- **text-indent:**
    1. lenght: uma medida reconhecida pelas CSS (px, pt, em, cm, ...)
    2. % : porcentagem da largura do elemento pai
- **text-transform:**
    1. none: texto normal
    2. capitalize: todas as primeiras letras do texto em maiúsculas
    3. uppercase: todas as letras do texto em maiúsculas
    4. lowercase: todas as letras do texto em minúsculas
- **direction:**
    1. ltr: texto escrito da esquerda para a direita
    2. rtl: texto escrito da direita para a esquerda
- **white-space:**
    1. normal: os espaços em branco serão ignorados pelo browser
    2. pre: os espaços em branco serão preservados pelo browser
    3. nowrap: o texto será apresentado todo ele numa linha única na tela. Não há quebra de linha até ser encontrada uma tag

Vamos a seguir analisar cada uma delas detalhadamente através de exemplos práticos.

Como estudar e entender os exemplos

Para cada propriedade apresento as regras CSS para um ou mais elementos HTML e definidas dentro de uma folha de estilos incorporada, bem como um trecho do documento HTML onde se aplicam as regras.

A seguir mostro o efeito que a regra produz. Observe a regra e o efeito e para melhor fixar seu aprendizado reproduza o código no seu editor, mude os valores e veja o resultado no browser. Esta é a melhor e mais rápida maneira de você aprender CSS. Bons estudos! E faça ótimo proveito dos tutoriais.

color ... A cor do texto

```
<html>
<head>
<style type="text/css">
<!--
h1 {color: #FF0000;}
h2 {color: #00FF00;}
p {color: rgb(0,0,255);}
-->
</style>
</head>
```

```
<body>
<h1>Este  cabeçalho é vermelho</h1>
<h2>Este cabeçalho é verde</h2>
<p>Este parágrafo é azul</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Este cabeçalho é vermelho

Este cabeçalho é verde

Este parágrafo é azul

letter-spacing...O espaço entre letras

```
<html>
<head>
<style type="text/css">
<!--
h2 {letter-spacing: 1.2em;}
p {letter-spacing: 0.4cm;}
-->
</style>
</head>
<body>
<h2> Este é o cabeçalho</h2>
<p> Este é o parágrafo</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

E s t e é o c a b e ç a l h o

E s t e é o p a r a g r á f o

word-spacing...O espaço entre palavras

```
<html>
<head>
<style type="text/css">
<!--
h2 {word-spacing: 1.8em;}
p {word-spacing: 80px;}
-->
</style>
</head>
```

```
<body>
<h2> Este é o cabeçalho</h2>
<p> Este é o parágrafo</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Este é o cabeçalho

Este é o paragráfo

text-align...Alinhar o texto

```
<html>
<head>
<style type="text/css">
<!--
h1 {text-align: left;}
h2 {text-align: center;}
h3 {text-align: right;}
p {text-align: justify;}
-->
</style>
</head>
<body>
<h1>Este é o cabeçalho 1</h1>
<h2>Este é o cabeçalho 2</h2>
<h3>Este é o cabeçalho 3</h3>
<p>Este é o parágrafo cujo texto ...</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Este é o cabeçalho 1

Este é o cabeçalho 2

Este é o cabeçalho 3

Este é o parágrafo cujo texto foi alongado para mais de duas linhas para que você possa visualizar o efeito de `text-align: justify` que força o texto a estender-se desde a direita até a esquerda.

text-decoration...Decoração do texto

```
<html>
<head>
<style type="text/css">
<!--
h1 {text-decoration: underline;}
h2 {text-decoration: line-through;}
h3 {text-decoration: overline;}
a {text-decoration: none;}
-->
</style>
</head>
<body>
<h1>Texto com sublinhado</h1>
<h2>Texto com linha em cima</h2>
<h3>Texto com sobrelinhado</h3>
<p>
<a                                                    href="http://www.maujor.com">
Este é um link sem sublinhado</a>
</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Texto com sublinhado

~~Texto com linha em cima~~

Texto com sobrelinhado

Este é um link sem sublinhado

text-indent...Recuo do texto

```
<html>
<head>
<style type="text/css">
<!--
h3 {text-indent: 80px;}
p {text-indent: 3em;}
-->
</style>
</head>
<body>
<h3>Texto com recuo de 80 pixel</h3>
<p>Texto com recuo de 3.0em</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Texto com recuo de 80 pixeis

Texto com recuo de 3.0em

text-transform...Forma das letras do texto

```
<html>
<head>
<style type="text/css">
<!--
h1 {text-transform: none;}
h2 {text-transform: capitalize;}
h3 {text-transform: uppercase;}
h4 {text-transform: lowercase;}
-->
</style>
</head>
<body>
<h1>Texto com letras como digitadas</h1>
<h2>Texto com primeira letra das palavras, maiúsculas</h2>
<h3>Texto com todas letras, maiúsculas</h3>
<h4>Texto com letras minúsculas</h4>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Texto com letras como digitadas

Texto com primeira letra das palavras, maiúsculas

TEXTO COM TODAS LETRAS, MAIÚSCULAS

Texto com letras minúsculas

Você poder fazer uso de um excelente editor para a propriedade background e descobrir mais efeitos para complementar este tutorial, nesta página interativa.

A propriedade margin

As margens nos elementos HTML

A propriedade para margens, define um valor para espessura das margens dos elementos HTML.

As propriedades para margens são as listadas abaixo:

- margin-top.....define a margem superior;
- margin-right.....define a margem direita;
- margin-bottom.....define a margem inferior;
- margin-left.....define a margem esquerda;
- margin.....**maneira abreviada para todas as margens**

Valores válidos para a propriedade `margin`

1. auto: valor default da margem
2. length: uma medida reconhecida pelas CSS (px, pt, em, cm, ...)
3. %: porcentagem da largura do elemento pai

Vamos a seguir analisar cada uma delas detalhadamente através de exemplos práticos.

Como estudar e entender os exemplos

Para cada propriedade apresento as regras CSS válidas para um elemento HTML, e, definidas dentro de uma folha de estilos incorporada, bem como um trecho do documento HTML onde se aplicam as regras.

A seguir mostro o efeito que a regra produz. Observe a regra e o efeito e para melhor fixar seu aprendizado reproduza o código no seu editor, mude os valores e veja o resultado no browser. Bons estudos! E faça ótimo proveito do tutorial.

Nota: Coloquei um fundo cinza mais escuro nos exemplos para facilitar a visualização.

`margin-top`...a margem superior

```
<html>
<head>
<style type="text/css">
<!--
p {margin-top: 2cm;}
-->
</style>
</head>
<body>
<p>Uma margem superior de 2cm</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Uma margem superior de 2cm

`margin-right`...a margem direita

```
<html>
<head>
<style type="text/css">
<!--
p {margin-right: 300px;}
-->
</style>
</head>
<body>
<p>Uma margem direita de 300px nesta
 frase mais longa dentro do parágrafo</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Uma margem direita de 300px nesta frase mais longa

dentro do parágrafo

`margin-bottom`...a margem inferior

```
<html>
<head>
<style type="text/css">
<!--
p {margin-bottom: 2em;}
-->
</style>
</head>
<body>
<p>Uma margem inferior de 2.0em</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Uma margem inferior de 2.0em

`margin-left`...a margem esquerda

```
<html>
<head>
<style type="text/css">
<!--
p {margin-left: 10%;}
-->
</style>
</head>
<body>
<p>Uma margem esquerda de 10%</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Uma margem esquerda de 10%

`margin`...todas as quatro margens em uma declaração única

A propriedade da margin permitem que você controle o espaçamento em volta dos elementos HTML. São válidos **valores negativos** para margem, com o objetivo de sobrepor elementos.

Em declaração única a ordem das margens é: **superior, direita, inferior** e **esquerda**.

Há quatro modos de se declarar abreviadamente as margens:

1. `margin`: valor1.....as 4 margens terão valor1;
2. `margin`: valor1 valor2.....margem superior e inferior terão valor1 - margem direita e esquerda terão valor2
3. `margin`: valor1 valor2 valor3.....margem superior terá valor1 - margem direita e esquerda terão valor2 - margem inferior terá valor3
4. `margin`: valor1 valor2 valor3 valor4....margens superior, direita, inferior e esquerda nesta ordem.

```
<html>
<head>
```

```
<style type="text/css">
<!--
p {margin: 20px 40px 80px 5px;}
-->
</style>
</head>
<body>
<p>Uma margem superior de 20px, uma margem direita de 40px,
uma margem inferior de 80px e uma margem esquerda de 5px</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Uma margem superior de 20px, uma margem direita de 40px, uma margem inferior de 80px e uma margem esquerda de 5px

A propriedade border

As bordas nos elementos HTML

As propriedades para as bordas, definem as características (os valores na regra CSS) das quatro bordas de um elemento HTML.

As propriedades para as bordas são as listadas abaixo:

- border-width:.....espessura da borda
- border-style:.....estilo da borda
- border-color:.....cor da borda
- ----------
- border-top-width:.....espessura da borda superior
- border-top-style:.....estilo da borda superior
- border-top-color:.....cor da borda superior
- ----------
- border-right-width:.....espessura da borda direita
- border-right-style:.....estilo da borda direita
- border-right-color:.....cor da borda direita
- ----------
- border-bottom-width:.....espessura da borda inferior
- border-bottom-style:.....estilo da borda inferior
- border-bottom-color:.....cor da borda inferior

- ----------
- border-left-width:.....espessura da borda esquerda
- border-left-style:.....estilo da borda esquerda
- border-left-color:.....cor da borda esquerda
- ----------
- border-top:...**maneira abreviada para todas as propriedades da borda superior**
- border-right:..**maneira abreviada para todas as propriedades da borda direita**
- border-bottom:..**maneira abreviada para todas as propriedades da borda inferior**
- border-left:..**maneira abreviada para todas as propriedades da borda esquerda**
- border:.....**maneira abreviada para todas as quatro bordas**

Valores válidos para as propriedades das bordas

- **color:**
    1. código hexadecimal: #FFFFFF
    2. código rgb: rgb(255,235,0)
    3. nome da cor: red, blue, green...etc
- **style:**
    1. none: nenhuma borda
    2. hidden: equivalente a none
    3. dotted: borda pontilhada
    4. dashed: borda tracejada
    5. solid: borda contínua
    6. double: borda dupla
    7. groove: borda entalhada
    8. ridge: borda em ressalto
    9. inset: borda em baixo relevo
    10. outset: borda em alto relevo
- **width:**
    1. thin: borda fina
    2. medium: borda média
    3. thick: borda grossa

4. length: uma medida reconhecida pelas CSS (px, pt, em, cm, ...)

Vamos a seguir analisar algumas delas detalhadamente através de exemplos práticos.

Como estudar e entender os exemplos

Para cada propriedade apresento as regras CSS para um ou mais elementos HTML e definidas dentro de uma folha de estilos incorporada, bem como um trecho do documento HTML onde se aplicam as regras.

A seguir mostro o efeito que a regra produz. Observe a regra e o efeito e para melhor fixar seu aprendizado reproduza o código no seu editor, mude os valores e veja o resultado no browser. Bons estudos! E faça ótimo proveito dos tutoriais.

border-width, border-style e border-color

```
<html>
<head>
<style type="text/css">
<!--
h3   {
border-width: medium;
border-style: solid;
border-color: #0000FF;
    }
p   {
border-width: 6px;
border-style: dashed;
border-color: #FF0000;
    }
-->
</style>
</head>
<body>
<h3>Borda média, contínua e azul</h3>
<p>Borda 6px, tracejada e vermelha</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Borda média, contínua e azul

Borda 6px, tracejada e vermelha

Nota: A propriedade `border-color` não é reconhecida pelo Internet Explorer se for usada isolada. Use a propriedade `border-style` para ser reconhecida pelo Internet Explorer.

Nota: A propriedade `border-color` não é reconhecida pelo Netscape. Use a propriedade `border` para ser reconhecida pelo Netscape.

border-style

Abaixo os estilos de bordas obtidos com a declaração `border-style: valor` (dotted, dashed, etc..)

Borda dotted

Borda dashed

Borda solid

Borda double

Borda groove

Borda ridge

Borda inset

Borda outset

border-width

Estude o código abaixo e tire suas conclusões de como obter outros efeitos com espessuras de bordas

```
<html>
<head>
<style type="text/css">
p    {
border-style: solid;
border-bottom-width: 10px;
border-top-width: 0px;
border-right-width: 0px;
border-left-width: 0px;
 }
</style>
</head>
```

```
<body>
<p>Borda com espessura inferior de 10px</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Borda com espessura inferior de 10px

Nota: A propriedade `border-bottom-width` não é reconhecida pelo Internet Explorer se usada isoladamente. Use `border-style` para ser reconhecida pelo Internet Explorer.

Definir a espessura das bordas superior, esquerda e direita

Proceda de modo semelhante ao mostrado acima.

Erro! Indicador não definido.

border (declaração única)

Esta é a maneira abreviada de você escrever uma regra para as propriedades das bordas.

Você pode declarar todas as tres propriedadesdas bordas em uma regra única:

A sintaxe geral é esta: `border: size style color;` em qualquer ordem. **Nota: Aconselho a escolher, e adotar, sempre a mesma ordem.**

```
<html>
<head>
<style type="text/css">
<!--
p     {
border: thick  groove  rgb(255,0,255)
    }
</style>
</head>
<body>
<p>Bordas em declaração única</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Bordas em declaração única

Propriedades CSS das bordas

As propriedades das bordas permitem que você controle o estilo a cor e a espessura das bordas de um elemento HTML.
As propriedades são muitas e como você viu, podem ser declaradas para cada uma das quatro bordas individualmente.

Neste tutorial abordei sumariamente algumas das propriedades, fornecendo as bases para seus estudos mais completos.

Você poder fazer uso de um excelente editor de bordas para descobrir mais efeitos e complementar este tutorial, nesta página interativa.

A propriedade padding

Os espaçamentos nos elementos HTML

A propriedade para espaçamentos (alguns traduzem como "enchimento"), define um valor para os espaçamentos entre o conteúdo e as bordas dos elementos HTML.

As propriedades para espaçamentos são as listadas abaixo:

- padding-top.....define a espaçamento superior;
- padding-right.....define a espaçamento direita;
- padding-bottom.....define a espaçamento inferior;
- padding-left.....define a espaçamento esquerda;
- padding.....**maneira abreviada para todas os espaçamentos**

Valores válidos para as propriedades de espaçamento

1. auto: valor default da margem
2. length: uma medida reconhecida pelas CSS (px, pt, em, cm, ...)
3. %: porcentagem da largura do elemento pai

Vamos a seguir analisar cada uma delas detalhadamente através de exemplos práticos.

Como estudar e entender os exemplos

Para cada propriedade apresento as regras CSS para um elemento HTML e definidas dentro de uma folha de estilos incorporada, bem como um trecho do documento HTML onde se aplicam as regras.

A seguir mostro o efeito que a regra produz. Observe a regra e o efeito e para melhor fixar seu aprendizado reproduza o código no seu editor, mude os valores e veja o resultado no browser. Bons estudos! E faça ótimo proveito dos tutorial.

Nota: Coloquei um fundo cinza mais escuro nos exemplos para facilitar a visualização.

`padding-top`...o espaçamento superior

```
<html>
<head>
<style type="text/css">
<!--
p {padding-top: 2cm;}
-->
</style>
</head>
<body>
<p>Um espaçamento superior de 2cm</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Um espaçamento superior de 2cm

`padding-right`...o espaçamento direito

```
<html>
<head>
<style type="text/css">
<!--
p {padding-right: 300px;}
-->
</style>
</head>
<body>
<p>Um espaçamento direito de 300px nesta
 frase mais longa dentro do parágrafo</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Um espaçamento a direita de 300px nesta frase mais longa dentro do parágrafo

`padding-bottom`...o espaçamento inferior

```
<html>
<head>
<style type="text/css">
<!--
p {padding-bottom: 2em;}
-->
</style>
</head>
<body>
<p>Um espaçamento inferior de 2.0em</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Um espaçamento inferior de 2.0em

`padding-left`...o espaçamento esquerdo

```
<html>
<head>
<style type="text/css">
<!--
```

```
p {padding-left: 10%;}
-->
</style>
</head>
<body>
<p>Um espaçamento esquerdo de 10%</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Um espaçamento esquerdo de 10%

`padding`...todos os quatro espaçamentos em uma declaração única

A propriedade padding permite que você controle o espaçamento entre o conteúdo e as bordas dos elementos HTML.

Não são válidos valores negativos para espaçamento.

Em declaração única a ordem das espaçamentos é: **superior, direito, inferior** e **esquerdo**.

Há quatro modos de se declarar abreviadamente os espaçamentos:

1. padding: valor1.....os 4 espaçamentos terão valor1;
2. padding: valor1 valor2.....espaçamento superior e inferior terão valor1 - espaçamento direito e esquerdo terão valor2
3. padding: valor1 valor2 valor3.....espaçamento superior terá valor1 - espaçamento direito e esquerdo terão valor2 - espaçamento inferior terá valor3
4. padding: valor1 valor2 valor3 valor4....os espaçamentos superior, direito, inferior e esquerdo nesta ordem.

```
<html>
<head>
<style type="text/css">
<!--
p {padding: 20px 40px 80px 5px;}
-->
</style>
</head>
<body>
<p>Um espaçamento superior de 20px,
 um espaçamento direito de 40px,
um espaçamento inferior de 80px
e um espaçamento esquerdo de 5px</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Um espaçamento superior de 20px, um espaçamento direito de 40px, um espaçamento inferior de 80px e um espaçamento esquerdo de 5px

A propriedade background

O fundo dos elementos HTML

A propriedade background define as características (os valores na regra CSS) do fundo dos elementos HTML.

As propriedades background são as listadas abaixo:

- background-color..... cor do fundo;
- background-image..... imagem de fundo;
- background-repeat..... maneira como a imagem de fundo é posicionada;
- background-attachment.....se a imagem de fundo "rola" ou não com a tela;
- background-position.....como e onde a imagem de fundo é posicionada;
- background.....**maneira abreviada para todas as propriedades**;

Valores válidos para as propriedades do fundo

- **background-color:**
    1. código hexadecimal: #FFFFFF
    2. código rgb: rgb(255,235,0)
    3. nome da cor: red, blue, green...etc
    4. transparente: transparent
- **background-image:**
    1. URL: url(caminho/imagem.gif)
- **background-repeat:**
    1. não repete: no-repeat
    2. repete vertical e horizontal: repeat
    3. repete vertical: repeat-y
    4. repete horizontal: repeat-x
- **background-attachment:**
    1. imagem fixa na tela: fixed
    2. imagem "rola" com a tela: scroll
- **background-position:**

1. x-pos y-pos
2. x-% y-%
3. top left
4. top center
5. top right
6. center left
7. center center
8. center right
9. bottom left
10. bottom center
11. bottom right

Vamos a seguir analisar cada uma delas detalhadamente através de exemplos práticos.

Como estudar e entender os exemplos

Para cada propriedade apresento as regras CSS para um ou mais elementos HTML e definidas dentro de uma folha de estilos incorporada, bem como um trecho do documento HTML onde se aplicam as regras.

A seguir mostro o efeito que a regra produz. Observe a regra e o efeito e para melhor fixar seu aprendizado reproduza o código no seu editor, mude os valores e veja o resultado no browser. Esta é a melhor e mais rápida maneira de você aprender CSS. Bons estudos! E faça ótimo proveito dos tutoriais.

A cor do fundo

```
<html>
<head>
<style type="text/css">
<!--
body {background-color: #CCFFFF;} /*azul claro*/
h2 {background-color: #FF0000;}   /* vermelho */
p {background-color: #00FF00;}    /*  verde   */
-->
</style>
</head>
<body>
<h2>Estude CSS</h2>
<p>Com CSS você controla melhor seu layout</p>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Estude CSS

Com CSS você controla melhor seu layout

A imagem de fundo

```
<html>
<head>
<style type="text/css">
<!--
body { background-image: url("/images/css.gif");}
-->
</style>
</head>
<body>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Repetir verticalmente a imagem de fundo

```
<html>
<head>
<style type="text/css">
<!--
body {
background-image: url("/images/css.gif");
background-repeat: repeat-y;
}
-->
</style>
</head>
<body>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Repetir horizontalmente a imagem de fundo

```
<html>
<head>
style type="text/css">
<!--
body {
background-image: url("/images/css.gif");
background-repeat: repeat-x;
}
-->
</style
</head>
<body>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

Posicionar uma imagem de fundo

```
<html>
<head>
<style type="text/css">
<!--
body {
background-image: url("/images/css.gif");
background-repeat: no-repeat;
background-position: 200px 70px;
}
-->
</style>
</head>
<body>
</body>
</html>
```

Este é o efeito da folha de estilo acima: a imagem esta posicionada a 200 pixel da margem esquerda e 70 pixel da margem superior

Ajustar uma imagem de fundo fixa, que não "rola" com a tela.

```
<html>
<head>
<style type="text/css">
<!--
body {
background-image: url("/images/css.gif");
background-repeat: no-repeat;
background-attachment: fixed;
}
-->
</style>
</head>
<body>
</body>
</html>
```

Este é o efeito da aplicação das regras CSS acima em uma página web.

Todas as propriedades do fundo em uma declaração única

Esta é a maneira abreviada de você escrever uma regra para as propriedades do fundo.

Você pode declarar todas ou algumas das propriedades estudadas em uma regra única:

A sintaxe geral é esta: `background: color image repeat attachment position;` em qualquer ordem, podendo ser omitido um mais valores.

Veja o exemplo abaixo:

```
<html>
<head>>
<style type="text/css">/>
<!--
body {
background: #00FF00 url("css.gif")
no-repeat fixed 200px 70px;
}
-->
</style>
</head>
```

Você poder fazer uso de um excelente editor para a propriedade background e descobrir mais efeitos para complementar este tutorial, nesta página interativa.

A propriedade list

Mudando o estilo das listas HTML

A propriedade list define as características (valores) das listas HTML.

As propriedades list são as listadas abaixo:

- list-style-image..... imagem como marcador da lista;
- list-style-position.....onde o marcador da lista é posicionado;
- list-style-type.....tipo do marcador da lista;
- list-style.....**maneira abreviada para todas as propriedades**;

Valores válidos para as propriedades do lista

- `list-style-image`:
    1. none
    2. URL: url(caminho/marcador.gif)
- `list-style-position`:
    1. outside: marcador fora do alinhamento do texto
    2. inside: marcador alinhado com texto
- `list-style-type`:
    1. none: sem marcador
    2. disc: círculo (bolinha cheia)
    3. circle: circunferência (bolinha vazia)

4. square: quadrado cheio
5. decimal: números 1, 2, 3, 4, ...
6. decimal-leading-zero
7. lower-roman: romano minúsculo i, ii, iii, iv, ...
8. upper-roman: romano maiúsculo I, II, III, IV, ...
9. lower-alpha: letra minúscula a, b, c, d, ...
10. upper-alpha: letra maiúscula A, B, C, D, ...
11. lower-greek
12. lower-latin
13. upper-latin
14. hebrew
15. armenian
16. georgian
17. cjk-ideographic
18. hiragana
19. katakana
20. hiragana-iroha
21. katakana-iroha

Os tipos de 11 a 20 são de uso específico e sem suporte total pelos navegadores atuais e não serão tratados neste tutorial.

Vamos a seguir analisar cada uma delas detalhadamente através de exemplos práticos.

Como estudar e entender os exemplos

Para cada propriedade apresento as regras CSS para o elemento lista HTML e definidas dentro de uma folha de estilos incorporada, bem como um trecho do documento HTML onde se aplicam as regras.

A seguir mostro o efeito que a regra produz. Observe a regra e o efeito e para melhor fixar seu aprendizado reproduza o código no seu editor, mude os valores e veja o resultado no browser. Bons estudos! E faça ótimo proveito do tutorial.

`list-style-image`...imagem para marcadores de lista

Este exemplo demonstra como definir uma imagem de marcador de listas

```
<html>
<head>
<style type="text/css">
<!--
ul
{
list-style-image: url("seta.gif");
}
-->
</style>
</head>
<body>
<ul>
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ul>
</body>
</html>
```

A folha de estilo acima resultará nesta lista:

- Item um
- Item dois
- Item tres

`list-style-position`...posição dos marcadores de lista

Este exemplo demonstra como posicionar os marcadores de listas

```
html>
<head>
<style type="text/css">
<!--
ul.inside
{
list-style-position: inside;
}
ul.outside
{
list-style-position: outside;
}
-->
</style>
</head>
<body>
<ul class="inside">
<li>Este texto destina-se a demonstrar o
valor: "inside" dos marcadores de listas.</li>
<li>E aqui continuamos com mais texto para
 fixar o valor:"inside" dosmarcadores de listas.</li>
</ul>
<ul class="outside">
```

```
<li>Este texto destina-se a demonstrar o
 valor: "outside" dos marcadores de listas.</li>
<li>E aqui continuamos com mais texto para
 fixar o valor:"outside" dos marcadores de listas.</li>
</ul>
</body>
</html>
```

A folha de estilo acima resultará nesta lista:

- Este texto destina-se a demonstrar o valor: "inside" dos marcadores
- E aqui continuamos com mais texto para fixar o valor:"inside" dos marcadores de listas.
- Este texto destina-se a demonstrar o valor: "outside" dos marcadores
- E aqui continuamos com mais texto para fixar o valor:"outside" dos marcadores de listas.

`list-style-type`...os tipos de marcadores de lista

Definir os marcadores de listas não ordenadas

Este exemplo demonstra como definir os marcadores de listas não ordenadas.

```
<html>
<head>
<style type="text/css">
<!--
ul.none {
list-style-type: none;
}
ul.disc {
list-style-type: disc;
}
ul.circle {
list-style-type: circle;
}
ul.square {
list-style-type: square;
}
-->
</style>
</head>
<body>
<ul class="none">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ul>
<ul class="disc">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ul>
<ul class="circle">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ul>
```

```
<ul class="square">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ul>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

- Item um
- Item dois
- Item tres
- Item um
- Item dois
- Item tres
- Item um
- Item dois
- Item tres
- Item um
- Item dois
- Item tres

Definir os marcadores de listas ordenadas

Este exemplo demonstra como definir os marcadores de listas ordenadas.

```
<html>
<head>
<style type="text/css">
<!--
ol.decimal
{
list-style-type: decimal;
}
ol.lroman
{
list-style-type: lower-roman;
}
ol.uroman
{
list-style-type: upper-roman;
}
ol.lalpha
{
list-style-type: lower-alpha;
}
ol.ualpha
```

```
{
list-style-type: upper-alpha;
}
-->
</style>
</head>
<body>
<ol class="decimal">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ol>
<ol class="lroman">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ol>
<ol class="uroman">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ol>
<ol class="lalpha">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ol>
<ol class="ualpha">
<li>Item um</li>
<li>Item dois</li>
<li>Item tres</li>
</ol>
</body>
</html>
```

Este é o efeito da folha de estilo acima:

1. Item um
2. Item dois
3. Item tres

1. Item um
2. Item dois
3. Item tres

1. Item um
2. Item dois
3. Item tres

1. Item um
2. Item dois
3. Item tres

1. Item um
2. Item dois
3. Item tres

`list-style`...duas propriedades das listas em uma declaração única

Esta é a maneira abreviada de você escrever uma regra para as propriedades das listas.

Você pode declarar duas das propriedades estudadas em uma regra única:

A sintaxe geral é esta: `list-style: position; imagem` ou `list-style: position; type` podendo inverter a ordem.

Veja o exemplo abaixo:

```
<html>
<head>
<style type="text/css">
<!--
ul
{
list-style: inside url("seta.gif");
}
-->
</style>
</head>
<body>
<ul>
<li>Texto para demonstrar a propriedade
 de declaração única  para listas usando
CSS - Folhas de Estilo em Cascata;</li>
<li>Item dois;</li>
<li>Item tres.</li>
</ul>
</body>
</html>
```

A folha de estilo acima resultará nesta lista:

- Texto para demonstrar a propriedade de declaração única para listas usando CSS - Folhas de Estilo em Cascata;
- Item dois;
- Item tres.

Pseudo-elementos CSS

Sintaxe

São usados em CSS, para adicionar efeitos a um seletor, ou a parte de um seletor.

A sintaxe dos pseudo-elementos:

```
seletor:pseudo-elemento {propriedade: valor;}
```

As classes em CSS podem também ser usadas com pseudo-elementos.

Esta regra permite que você defina diferentes efeitos para pseudo-elementos localizados em diferentes lugares em uma mesma página.

```
seletor.classe:pseudo-elemento {propriedade: valor;}
```

O pseudo-elemento `first-letter`

O pseudo-elemento `first-letter` é usado para obter um efeito especial na **primeira letra** de um texto.

```
<html>
<head>
<style type="text/css">
p {
font-size: 12pt
}
p:first-letter {
font-size:300%;
}
</style>
</head>
<body>
<p>Este texto destina-se a demonstrar o
 pseudo-elemento first-letter, bla...bla...bla...
bla... bla...bla...bla...bla...bla... bla...bla...
bla... bla...bla...bla...bla...bla... bla...bla...</p>
</body>
</html>
```

O código acima produzirá esse efeito

Este texto destina-se a demosnstrar o pseudo-elemento first-letter, bla...bla...bla... bla... bla...bla...bla...bla...bla... bla...bla... bla... bla...bla...bla...bla...bla... bla...bla... bla... bla...bla...bla...bla...bla... bla...bla... bla... bla...bla...bla...bla...bla... bla...bla...

O pseudo-elemento `first-letter` somente pode ser usado com elementos de bloco.

Propriedades aplicáveis ao pseudo-elemento `first-letter`

- font - propriedades de letras
- color - propriedades de cores
- background - propriedades de fundo
- margin - propriedades de margens
- padding - propriedades de espaçamentos
- border - propriedades de bordas
- text-decoration
- vertical-align (somente para "float: none)
- text-transform

- line-height
- float
- clear

O pseudo-elemento `first-line`

O pseudo-elemento `first-line` é usado para obter um efeito especial na **primeira linha** de um texto.

```
<html>
<head>
<style type="text/css">
p {
font-size: 12pt
}
p:first-line {
color: #0000FF;
font-variant: small-caps;
}
</style>
</head>
<body>
<p>Um texto qualquer dentro
 de um pseudo-elemento first-line,
 para um  efeito especial na primeira linha</p>
</body>
</html>
```

O código acima produzirá esse efeito

Um texto qualquer dentro de um pseudo-elemento first-line, para um efeito especial na primeira linha. Notar a mudança de cor e o tipo de letra small-caps na primeira linha.

No exemplo acima toda a primeira linha sofre o efeito da definição do pseudo-elemento. A "quebra" da linha depente do tamanho da janela do browser.

O pseudo-elemento `first-line` somente pode ser usado com elementos de bloco.

Propriedades aplicáveis ao pseudo-elemento `first-line`

- font - propriedades de letras
- color - propriedades de cores
- background - propriedades de fundo
- word-spacing - espaçamento entre palavras
- letter-spacing - espaçamento entre letras
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

Pseudo-elementos em classes CSS

Pseudo-elementos podem ser combinados com classes CSS

```
<html>
<head>
<style type="text/css">
p.combinado:first-letter {
color: #FF0000;
font-size:xx-large;
}
</style>
</head>
<body>
<p class="combinado"> Uma frase com efeito
 especial combinado </p>
</body>
</html>
```

O código acima produzirá esse efeito

Uma frase com efeito especial combinado

Controlando as entrelinhas e o espaçamento entre elementos HTML

As propriedades `line-height` e `margin`

A propriedade CSS de dimensionamento `line-height` permite controlar o espaçamento entre linhas e a propriedade CSS `margin` permite controlar o espaçamento entre elementos HTML.

Observe abaixo o código HTML para um texto composto de dois parágrafos:

```
<html>
<head>
</head>
<body>

<p>
1o. Parágrafo....Lorem ipsum dolor sit
amet, consectetuer adipiscing elit.
 Nulla pharetra egestas neque.
Duis dolor lacus, volutpat ac,
vestibulum nec, suscipit a, felis.
Aenean pharetra orci id elit.
Duis non dui. Suspendisse potenti.
Ut ac risus. Etiam dignissim.
Quisque nec felis.
</p>

<p>
2o.Parágrafo.......Sed blandit est non
ante. Ut imperdiet sagittis mi.
Sed gravida sodales nisl. Ut hendrerit
ipsum eu enim. Duis tempus consequat mauris.
In hac habitasse platea dictumst.
Vivamus lectus justo, commodo in, rutrum non,
eleifend eget, pede. Sed ac lacus. In tortor.
</p>

</body>
```

```
</html>
```

O código acima é renderizado pelo navegador conforme mostrado abaixo.

Notar a distância entre as linhas em cada parágrafo, ou seja as entrelinhas (não confunda com distância entre parágrafos):

1o. Parágrafo....Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla pharetra egestas neque. Duis dolor lacus, volutpat ac, vestibulum nec, suscipit a, felis. Aenean pharetra orci id elit. Duis non dui. Suspendisse potenti. Ut ac risus. Etiam dignissim. Quisque nec felis.

2o.Parágrafo.....Sed blandit est non ante. Ut imperdiet sagittis mi. Sed gravida sodales nisl. Ut hendrerit ipsum eu enim. Duis tempus consequat mauris. In hac habitasse platea dictumst. Vivamus lectus justo, commodo in, rutrum non, eleifend eget, pede. Sed ac lacus. In tortor.

Alterando o espaçamento entre linhas

No código HTML mostrado acima vamos inserir uma regra CSS para `line-height` que é a propriedade CSS que controla as entrelinhas. Observe abaixo o mesmo código com a regra, definindo uma entrelinha igual a 200%.

Nota: A entrelinha default do bronwser é 100%.

Você pode usar qualquer medida de comprimento, válida em CSS (px, cm, em, %, in...) para o valor da propriedade `line-height`.

```
<html>
<head>
<style type="text/css">
<!--
p {
line-height:200%;
}
-->
</style>

</head>
<body>

<p>
1o. Parágrafo....Lorem ipsum dolor sit
amet, consectetuer adipiscing elit.
 Nulla pharetra egestas neque.
Duis dolor lacus, volutpat ac,
vestibulum nec, suscipit a, felis.
Aenean pharetra orci id elit.
Duis non dui. Suspendisse potenti.
Ut ac risus. Etiam dignissim.
Quisque nec felis.
</p>
```

```
<p>
2o.Parágrafo.......Sed blandit est non
ante. Ut imperdiet sagittis mi.
Sed gravida sodales nisl. Ut hendrerit
ipsum eu enim. Duis tempus consequat mauris.
In hac habitasse platea dictumst.
Vivamus lectus justo, commodo in, rutrum non,
eleifend eget, pede. Sed ac lacus. In tortor.
</p>

</body>
</html>
```

O código acima é renderizado pelo navegador conforme mostrado abaixo.

Notar que a entrelinha que era default 100%, agora está 200% ou seja dobrou:

Nota: Faça algumas experiências com o valor de `line-height`, usando inclusive valores abaixo de 100% e também outras medidas válidas (por exemplo: 12px, 2.3em, 3cm...etc...) e você vai constatar que tem o controle total das entrelinhas.

1o. Parágrafo....Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla pharetra egestas neque. Duis dolor lacus, volutpat ac, vestibulum nec, suscipit a, felis. Aenean pharetra orci id elit. Duis non dui. Suspendisse potenti. Ut ac risus. Etiam dignissim. Quisque nec felis.

2o.Parágrafo.....Sed blandit est non ante. Ut imperdiet sagittis mi. Sed gravida sodales nisl. Ut hendrerit ipsum eu enim. Duis tempus consequat mauris. In hac habitasse platea dictumst. Vivamus lectus justo, commodo in, rutrum non, eleifend eget, pede. Sed ac lacus. In tortor.

E o espaçamento (a distância) entre os parágrafos?

Aqui também o controle é todo seu via CSS.

E quem dita as regras para este espaçamento é a propriedade `margin`.

Vamos acresentar mais uma regra CSS no nosso código.

Se voce não lembra da propriedade `margin`, leia este tutorial sobre margens

```
<html>
<head>
<style type="text/css">
<!--
p {
line-height:200%;
margin: 40px 0  40px 0;
-->
</style>

</head>
<body>

<p>
1o. Parágrafo....Lorem ipsum dolor sit
amet, consectetuer adipiscing elit.
 Nulla pharetra egestas neque.
Duis dolor lacus, volutpat ac,
vestibulum nec, suscipit a, felis.
Aenean pharetra orci id elit.
Duis non dui. Suspendisse potenti.
Ut ac risus. Etiam dignissim.
Quisque nec felis.
</p>

<p>
2o.Parágrafo.......Sed blandit est non
ante. Ut imperdiet sagittis mi.
Sed gravida sodales nisl. Ut hendrerit
ipsum eu enim. Duis tempus consequat mauris.
In hac habitasse platea dictumst.
Vivamus lectus justo, commodo in, rutrum non,
eleifend eget, pede. Sed ac lacus. In tortor.
</p>

</body>
</html>
```

O código acima é renderizado pelo navegador conforme mostrado abaixo.

Notar que a entrelinha continua em 200% e agora o espaçamento entre parágrafos cresceu para 40 pixels, cumprindo a regra CSS, escrita.

1o. Parágrafo....Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nulla pharetra egestas neque.

Duis dolor lacus, volutpat ac, vestibulum nec, suscipit a, felis. Aenean pharetra orci id elit. Duis non dui.

Suspendisse potenti. Ut ac risus. Etiam dignissim. Quisque nec felis.

2o.Parágrafo.....Sed blandit est non ante. Ut imperdiet sagittis mi. Sed gravida sodales nisl. Ut hendrerit

ipsum eu enim. Duis tempus consequat mauris. In hac habitasse platea dictumst. Vivamus lectus justo, commodo in, rutrum non, eleifend eget, pede. Sed ac lacus. In tortor.

Você deve ter notado que o espaçamento do 1o. parágrafo para a borda superior do quadro amarelo e também a do 2o. parágrafo para a borda inferior do quadro amarelo, ambas AUMENTARAM.

Sim, este aumento no espaçamento cumpriu o prescrito na nova regra, ou seja: 40 pixel de margem superior e 40 pixel de margem inferior nos parágrafos.

Mas lembre-se o controle é SEU. Tem como evitar este espaçamento não previsto :-) Veja o item 1-) abaixo.

Dicas adicionais

1-) Para evitar aquele espaçamento referido acima, crie e aplique uma classe no parágrafo superior com `margin-top: 0;` (ou n pixels) e outra classe ao parágrafo inferior com `margin-bottom: 0;` (ou n pixels);

Você pode também declarar: `margin: 0 0 40px 0;` e suprimir o espaçamento superior, ou ainda `margin: 40px 0 0 0;` e suprimir o espaçamento inferior. E, uma série de outras combinações que ficam a título de exercícios para você.

2-) Se você deseja aplicar regras CSS em alguns elementos do documento e não em todos (por exemplo: alguns parágrafos na página seguirão uma regra `line-height` outros não) crie classes e aplique aos elementos.

As medidas CSS de comprimento

Introdução

Unidades de medida de comprimento CSS

As unidades de medida de comprimento CSS referem-se a medidas na horizontal ou na vertical (e em sentido mais amplo, em qualquer direção).

O formato para declarar o valor de uma unidade de medida CSS é um número com ou sem ponto decimal **imediatamente precedido** do sinal '+' (mais) ou do sinal '-' (menos), sendo o sinal '+' (mais) o valor "default" e **imediatamente seguido** por uma unidade identificadora (medida CSS válida - p.ex., px, em, deg, etc...). A unidade identificadora é opcional quando se declara um valor '0' (zero).

Algumas das propriedades CSS permitem que sejam declarados valores negativos para unidades de medida. A adoção de valores negativos podem complicar a formatação do elemento e devem ser usados com cautela. Se valores negativos não forem suportados pela aplicação de usuário, eles serão convertidos para o valor mais próximo suportado (e isso pode tornar-se desastroso para um layout).

Unidades de medida de comprimento CSS válidas

São dois os tipos de unidade de medida de comprimento CSS:

UNIDADE RELATIVA

- em
- ex
- px - pixel
- % - percentagem

as unidades relativas são referenciadas a outras unidades como veremos a seguir.

UNIDADE ABSOLUTA

- **pt - point :**1/72 in;
- **pc - pica :**12 points ou 1/6 in;
- **mm - milímetro :**1/10 cm;
- **cm - centímetro :**1/100 m;
- **in - polegada :**2,54 cm;

Unidade relativa- é aquela tomada em relação a uma outra medida. Folhas de Estilo em Cascata que usam unidades de comprimento relativas são mais apropriadas para ajustes de uso em diferentes tipos de mídia. (p. ex., de uma tela de monitor para uma impressora laser).

O valor é tomado em relação:

- **em**: ...ao tamanho da fonte ('font-size') herdada;
- **ex**: ...a altura da letra x (xis) da fonte herdada;
- **px**: ...ao dispositivo (midia) de exibição;
- **%**: ... a uma medida previamente definida.

Unidade absoluta - é aquela que não esta referenciada a qualquer outra unidade e nem é herdada. São unidades de medida de comprimento definidas nos sistemas de medidas pela física e em fim são os

conhecidos "centímetros, polegadas etc...). São indicadas para serem usadas quando as mídias de exibição são perfeitamente conhecidas.

Abaixo exemplos ilustrativos do uso destas medidas de comprimento CSS:

```
div { margin: 1.5em; }
h4 { margin: 2ex; }
p { font-size: 14px; }
.classe { padding: 90%; }
hr { width: 14pt; }
h1 { margin: 1pc; }
h2 { font-size: 4mm; }
p.classe { padding: 0.3cm; }
h5.classe { padding: 0.5in; }
```

Nota: Relembro que uma regra CSS tem a seguinte sintaxe

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

Entendendo as unidades de medida CSS

Vamos a seguir definir e analisar cada uma das unidades de medida CSS e apresentar exemplos práticos.

A unidade de medida - pixel

A unidade de medida de comprimento pixel é relativa a resolução do dispositivo de exibição (p.ex: a tela de um monitor).

Sem entrar em maiores considerações teóricas a mais simplista definição de pixel que encontrei é esta:

Pixel é o menor elemento em um dispositivo de exibição, ao qual é possivel atribuir-se uma cor.

Considere um dispositivo de exibição construido com uma densidade de 90 dpi (dpi = dots per inch = pontos por polegada). Por definição, a **referência padrão para pixel** é igual a um ponto no citado dispositivo. Daí pode-se concluir que **1 pixel** naquele dispositivo de exibição é igual a 1/90 inch = 0,28 mm.

Para uma densidade de 300 dpi 1 pixel é igual a 1/300 inch = 0,085mm

Assim, pixel é uma medida relativa a resolução do dispositivo de exibição.

A unidade de medida - em

A unidade de medida de comprimento **em** referencia-se ao tamanho da fonte (letra) do seletor onde for

declarada. Quando **em** for declarada para a propriedade `font-size` referencia-se ao tamanho da fonte (letra) do elemento pai. Quando **em** for declarada para o elemento raiz do documento (p. ex: `<html>` em documentos html) referencia-se ao valor inicial (default) do tamanho de fonte (letra).

Os exemplos abaixo esclarecem as definições:

```
h1 { line-height: 1.2em }
```

line-height de `<h1>` será 20% maior do que o tamanho das letras de `<h1>`

```
h1 { font-size: 1.2em }
```

font-size de `<h1>` será 20% maior do que o tamanho das letras herdado por `<h1>` p.ex.: se h1 estiver contido numa div com font-size=10px então font-size de h1 = 12px

A unidade de medida - ex

A unidade de medida de comprimento **ex** é igual a altura da letra **x**(xis) minúscula).

A unidade de medida - percentagem, %

Valores em percentagem são relativos a um outro valor anterior declarado. Este valor anterior há que estar bem definido e em geral esta definição está em uma determinada propriedade do mesmo elemento, na propriedade do elemento "pai" (por exemplo: uma medida CSS de comprimento) ou mesmo no contexto geral da formatação (por exemplo: a largura do bloco de conteúdo).

```
p { font-size: 10px }
p { line-height: 120% }/*120% de'font-size'=12px*/
```

Definindo cores em uma regra CSS

Objetivo

Detalhar as diferentes maneiras de se escrever a sintaxe para os valores das cores em uma regra CSS

Valores válidos para cores em CSS

Observe as regras de estilo a seguir:

```
1-) div.um {background-color: #FF0000;}

 2-) div.dois {background-color: #F00;}

 3-) div.tres {background-color: rgb(255, 0, 0);}

 4-) div.quatro {background-color: rgb(100%, 0%, 0%);}

 5-) div.cinco {background-color: red;}

 6-) div.seis {background-color: ThreeDShadow;}
```

Como você já deve ter concluido apresentei 06 (seis) maneiras diferentes de definir uma cor de fundo para uma `DIV` .

E, se considerarmos que para as duas primeiras regras é válido usar letras minúsculas, existem 08 (oito) maneiras de se definir uma cor em uma regra CSS.

As maneiras mais usadas são as mostradas em 1 e em 2, ou seja, com uso do código hexadecimal de cores.

O efeito das regras no navegador

Observe agora no screenshot a seguir como estas seis DIV's serão renderizadas.

As cinco primeiras estão com a mesma cor de fundo, vermelha o que nos leva a concluir que as cinco primeiras regras mostradas são equivalentes, ou seja são cinco maneiras diferentes de definir um mesmo valor para uma cor.

#FF0000 = #F00 = rgb(255,0,0) = rgb(100%,0%,0%) = red

A sexta cor, `ThreeDShadow` depende do equipamento do usuário.

Vejamos cada uma delas detalhadamente.

Definir uma cor pelo seu código hexadecimal

Esta é a maneira mais conhecida de definir uma cor.

Convém ressaltar que em uma regra CSS é indiferente usar letras maiúsculas ou minúsculas na sintaxe hexadecimal de cores e também que é válido abreviar a notação para três dígitos.

Na notação abreviada cada um dos três dígitos é automaticamente dobrado conforme exemplos a seguir:

```
#FFF = #FFFFFF

#CF9 = #CCFF99

#cde = #ccddee

#49c = #4499cc
```

Não é do escopo deste tutorial detalhar o código hexadecimal, contudo ressalto que os dezesseis dígitos hexadecimais são: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F e somente eles são válidos para definir uma cor, podendo em geral ser usada qualquer combinação deles. Assim: #FFDDHH não define uma cor, pois H não é válido.

Existem várias ferramentas online para determinar o código hexadecimal de uma cor. Uma das que eu costumo usar e indico para vocês é esta: http://www.colorschemer.com/online.html

Definir uma cor pelo seu código rgb

rgb é abreviatura para:
r = red (vermelha)
g = green (verde)
b = blue (azul)

Assim o código `rgb(xxx, yyy, zzz)` indica uma cor obtida com a mistura de uma quantidade xxx de vermelho com yyy de verde e com zzz de azul.

Duas são as maneiras de se definir a quantidade de cada uma das três cores:

Uma faixa de númeração de 0 (zero) até 255
Em percentagem de 0% até 100%

Não é válido usar em uma definição número e percentagem.

Exemplos:

definições válidas
`rgb(145, 230, 50) - rgb(20%, 0%, 70%)`

definição não válida

```
rgb(255, 20%, 120)
```

Não é do escopo deste tutorial detalhar as misturas de cor rgb.

No link indicado no item anterior é possível determinar também, o código rgb de uma cor

Definir cor por palavra-chave

Você pode definir uma cor usando o nome da cor. Os nomes de cor válidos são os listados nas recomendações CSS do W3C.

As Recomendações para CSS 2.1 listam as seguintes 17 cores:

aqua, black, blue, fuchsia, gray, green, lime, maroon, navy, olive, orange, purple, red, silver, teal, white, e yellow

Figura extraída das Recomendações para CSS2.1 no site do W3C http://www.w3.org

Assim, as regras a seguir são válidas para definir cor

```
p {color: aqua;}
div {background-color: teal;}
```

Definir cor baseado no sistema operacional do usuário

As recomendações para CSS2.1 preconizam a definição da cor baseado nas cores adotadas pelo sistema operacional do usuário.

Este tipo de unidade de definição de cor denominado System Colors está em desuso e não deverá constar das futuras Recomendações CSS3.

Trata-se de uma lista de nomes de cores válidas à semelhança da listagem de cores por palavra-chave e que se refere a áreas do sistema operacional.

As cores previstas são:

ActiveBorder, ActiveCaption, AppWorkspace, Background, ButtonFace, ButtonHighlight, ButtonShadow, ButtonText, CaptionText, GrayText, Highlight, HighlightText, InactiveBorder, InactiveCaption, InactiveCaptionText, InfoBackground, InfoText, Menu, MenuText, Scrollbar, ThreeDDarkShadow, ThreeDFace, ThreeDHighlight, ThreeDLightShadow, ThreeDShadow, Window, WindowFrame, WindowText

Embora os valores CSS sejam "case insensitives" recomenda-se usar a grafia com letras maiúsculas e minúsculas ao se escrever o nome das cores de sistema por razões de legibilidade.

Exemplos:

```
p                              {color:                              ThreeDLightShadow;}
div {background: ButtonShadow;}
```

Abreviando declarações e valores em regras CSS

Relembrando a sintaxe e a terminologia de uma regra CSS

É comum encontrar-se em muitos artigos sobre CSS escritos em blogs e sites, em textos de posts em fóruns, em listas de discussão e até mesmo em revistas e jornais, diferentes referências e denominações equivocadas para os componentes de uma regra CSS

Seletores são chamados de elementos ou de tags, *propriedades* são chamadas de seletores ou de atributos, *valores* são chamados de atributos ou de propriedades, *declarações* são chamadas de regras ou funções CSS e por aí vai em uma diversificada combinação dos termos acima citados em uma salada que acaba por confundir iniciantes e as vezes até mesmo outros já com alguma experiência com folhas de estilo em cascata.

Com a finalidade de facilitar o entendimento desta matéria e esclarecer a confusão que vem se formando em torno do assunto, vamos rever a sintaxe e a terminologia de uma regra CSS para que quando eu escrever seletor, declaração, propriedade e valor, não haja dúvidas sobre a porção da regra CSS a que estou me referindo e você não fique se perguntando onde estão os "atributos CSS, as tags CSS, e outros tantos termos equivocados".

Vejamos o que diz as Recomendações do W3C para Folhas de Estilo, nível 1 na seção intitulada Conceitos Básicos

“O projeto, ou desenho do layout, das folhas de estilos é fácil. É preciso apenas conhecer um pouco da linguagem HTML e possuir noções básicas dos termos usados em publicação eletrônica. Como exemplo, para ajustar a cor das letras de um elemento '`H1`' para azul, basta fazer:

```
H1 {color: blue}
```

Este exemplo mostra o que é uma 'regra' simples em CSS. Uma regra é composta de duas partes principais: um selector ('`H1`') e uma declaração ('`color: blue`'). Por sua vez, esta declaração também possui duas partes: uma propriedade ('`color`') e seu valor ('`blue`'). Embora este exemplo especifique apenas uma das várias propriedades necessárias para montar um documento HTML, ela constitui por si só uma 'folha de estilo'. Quando for combinada com outras folhas de estilo ela determinará a apresentação final do documento (uma característica fundamental é que as folhas de estilo podem ser combinadas).

O seletor funciona como a ponte de ligação entre o documento HTML e a folha de estilo, e todos os elementos HTML podem funcionar como possíveis seletores. Os vários elementos HTML estão definidos na Recomendação HTML

etc.”

A Recomendação do W3C define claramente que uma regra CSS é composta de um seletor e uma declaração e que a declaração compreende uma propriedade e um valor.

Na regra CSS a seguir:

```
H1 {color: blue}
```

a terminologia correta é:

- `H1` - seletor;
- `{color: blue}` - declaração;

- `color` - propriedade;
- `blue` - valor.

e a sintaxe correta é:

- Escrever o seletor e a seguir a declaração;
- A declaração deve estar entre { } (chaves);
- Na declaração, separar a propriedade e o valor por : (dois pontos);
- É permitido usar espaços em branco em qualquer quantidade entre cada um dos caracteres da regra;
- É permitido grupar declarações em uma mesma regra e neste caso as declarações deverão ser separadas por ; (ponto-e-vírgula) podendo todas elas estar em uma mesma linha ou em linhas distintas. É facultativo o uso de ; (ponto-e-vírgula) após a última declaração na regra;
- É indiferente o uso de maiúsculas e minúsculas em uma regra CSS, contudo as *classes* e *ID's* devem seguir a mesma grafia constante da marcação.

Estes são os termos normatizados de uma regra CSS e os que usaremos. Portanto, não existe "atributo CSS" ou "tag CSS" ou "elemento CSS" ou "função CSS" ou tantos outros equivocadamente escritos.

Não é do escopo deste tutorial detalhar as boas práticas de escrita das regras em uma folha de estilos. Sobre este assunto escrevi e recomendo a leitura do tutorial Dicas básicas para projetar folhas de estilos.

Abreviando valores de cores hexadecimais

O formato hexadecimal é uma das opções sintáticas mais usadas para se escrever o valor das cores em regras CSS. A regra a seguir define que os parágrafos serão na cor vermelha (#ff0000).

```
p {color: #ff0000;}
```

e que poderá ser abreviada para:

```
p {color: #f00;}
```

É válido abreviar cores hexadecimais para 3 dígitos. Valores escritos com 3 dígitos são interpretados como se cada um dos dígitos tivesse sido sido declarado duas vezes, isto é:

genericamente, #abc é equivalente #aabbcc

Exemplos:

```
#c30 = #cc3300
#999 = #999999
#ff0 = # ffff00
#d61 = #dd6611
```

É fácil concluir que a abreviação de cores hexadecimais somente é possível para as cores constituidas por 3 pares de dígitos hexadecimais.

Valores para os quatro lados de um elemento nível de bloco

Um elemento nível de bloco ou uma 'caixa' admite estilização em seus quatro lados para algumas propriedades como `border` e `padding` entre outras.
Por exemplo: você pode definir um `padding` superior, um `padding` à direita, um `padding` inferior e um

`padding` à esquerda para uma `div`.

A sequência em que você escreve os valores para estilizar os quatro lados de uma 'caixa' é rígida e fixa em uma regra CSS e obedece a seguinte ordem:

em cima , lado direito, embaixo, lado esquerdo

Faça uma analogia com o relógio para não esquecer a sequência. 12 horas (superior), 3 horas (direita), 6 horas (inferior), 9 horas (esquerda).

A regra `div {padding: 2px 3px 8px 7px;}` define para a `div`: um `padding` inferior igual a 8px: um `padding` superior igual a 2px; um `padding` à esquerda igual a 7px: um `padding` à direita igual a 3px.

Além da mostrada acima é válido abreviar declarações que envolvem os quatros lados de uma 'caixa' de outras 3 maneiras diferentes como mostradas a seguir:

1. `div {padding: 10px;}` `padding` de 10px para os 4 lados;
2. `div {padding: 6px 8px;}` `padding` de 6px para os lados superior e inferior e de 8px para os lados direito e esquedo;
3. `div {padding: 2px 4px 9px;}` `padding` de 2px para o lado superior, de 4px para os lados direito e esquerdo e de 9px para o lado inferior.

Propriedades que admitem abreviação

Veremos ao longo deste tutorial, como abreviar as seguintes propriedades CSS:

1. margin;
2. padding;
3. background;
4. font;
5. list;
6. outline;
7. border.

Abreviando `margin`

As regras a seguir definem valores para as 4 margens para uma `div`:

```
div {
margin-top:10px;
margin-right:8px;
margin-bottom:0;
margin-left:5px;
}
```

E pode ser abreviada para:

```
div {
margin:10px 8px 0 5px;
```

```
}
```

Abreviando `padding`

As regras a seguir definem valores para os 4 paddings de um parágrafo:

```
p {
padding-bottom:6px;
padding-top:12px;
padding-left:1px;
padding-right:2px;
}
```

E pode ser abreviada para:

```
div {
padding:12px 2px 6px 1px;
}
```

Abreviando `background`

As regras a seguir definem valores para propriedades background de uma `div`:

```
div {
background-color:#ffffcc;
background-image:url(fundo.gif);
background-repeat:no-repeat;
background-attachment:fixed;
background-position:20px 10px;
}
```

E pode ser abreviada para:

```
div {
background:#ffc url(fundo.gif) no-repeat fixed 20px 10px;
}
```

Abreviando `font`

As regras a seguir definem valores para propriedades de font em um documento:

```
body {
font-style:italic;
font-variant:small-caps;
font-weight:bold;
font-size:11px;
line-height:15px;
font-family:Arial, Helvetica, Sans-serif;
}
```

E pode ser abreviada para:

```
body {
font:italic small-caps bold 11px/15px Arial, Helvetica, Sans-serif;
}
```

Nota: Para abreviar a propriedade `font` é obrigatório definir no mínimo os valores de tamanho e família da `font`. Os demais valores são facultativos. A ordem de declaração dos valores é importante e deve ser assim:

1. começar com `font-style`, `font-variant` e `font-weight` sedo que estes três valores são facultativos e podem ser escritos em qualquer ordem;
2. a seguir declarar obrigatoriamente `font-size` e opcionalmente `line-height` (`font-size/line-height`);

3. finalmente declarar obrigatoriamente `font-family`.

Abreviando `list`

As regras a seguir definem valores para propriedades de listas:

```
ul {
list-style-type:square;
list-style-position:inside;
list-style-image:url(image.gif);
}
```

E pode ser abreviada para:

```
ul {list-style:square inside url(image.gif);}
```

A propriedade: `list-style-type` pode ser abreviada para **`list-style.`** Por exemplo: `list-style-type:none` pode ser abreviada para **`list-style:none;`**

Abreviando `outline`

A propriedade `outline` é pouco conhecida e empregada. Serve para colocar uma margem ao redor de um elemento, com a finalidade de destacá-lo no contexto. Difere da propriedade `border` por não interferir com as dimensões do box model, isto é, não ocupa espaço no box do elemento e em consequência não afeta o posicionamento do box e nem dos boxes adjacentes.

As regras a seguir definem a marcação de um 'destaque' em linha vermelha sólida de 1px ao redor do elemento `h2`:

```
h2 {
outline-color:#f00;
outline-style:solid;
outline-width:1px;
}
```

E pode ser abreviada para:

```
h2 {
outline:#f00 solid 1px;
}
```

Exemplo: Para este parágrafo eu defini um destaque (`outline`) de 5px em linha tracejada na cor azul que será visualizado nos Mozillas e no Ópera, mas não no Internet Explorer que não suporta `outline`.

Abreviando `border`

As regras a seguir definem valores para propriedades de borda:

```
div {
border-width:1px;
border-style:solid;
border-color:#f00;
}
```

E pode ser abreviada para:

```
div {border:1px solid #f00;}
```

Erro! Indicador não definido.

As regras a seguir definem valores para espessuras de borda:

```
p {
border-top-width:2px;
border-right-width:1px;
border-bottom-width:3px;
```

```
border-left-width:5px;
}
```

E pode ser abreviada para:

```
p {border-width:2px 1px 3px 5px;}
```

Erro! Indicador não definido.

As regras a seguir definem valores para cores de borda:

```
h1 {
border-top-color:#f00;
border-right-color:#ccc;
border-bottom-color:#00f;
border-left-color:#999;
}
```

E pode ser abreviada para:

```
p {border-color:#f00 #ccc #00f #999;}
```

Erro! Indicador não definido.

As regras a seguir definem valores para estilos de borda:

```
p {
border-top-style:solid;
border-right-style:ridge;
border-bottom-style:double;
border-left-style:dotted;
}
```

E pode ser abreviada para:

```
p {border-style:solid ridge double dotted;}
```

Introdução ao PHP

O que é PHP ?

PHP é acrônimo de Hypertext Preprocessor (pré-processador de hipertexto), uma poderosa linguagem de programação mundialmente usada principalmente no ambiente Web para gerar conteúdos dinâmicos na internet.

O PHP é um produto de código-fonte aberto, o que significa que você tem acesso ao seu código-fonte. É possível utilizá-lo, alterá-lo e redistribuí-lo sem pagar nada.

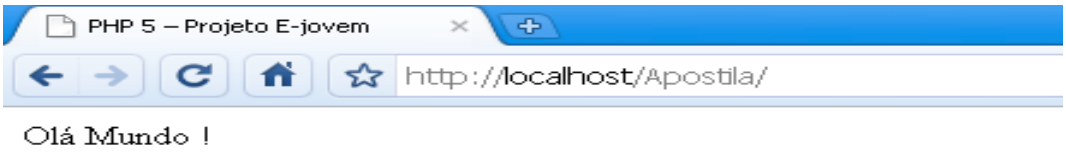
Dentro de uma página HTML, você pode embutir código PHP que será executado toda vez que a página for visitada. O código PHP é interpretado no servidor Web e gera HTML ou outra saída que o visitante verá.

Vejamos um exemplo da utilização do PHP:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
  <title>PHP 5 - Projeto E-jovem</title>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
   </head>
<body>
<?php
echo " Olá Mundo ! " ;
?>


    </body>
</html>
```

Este exemplo produz esta página HTML:



Como surgiu ?

O PHP foi concebido em 1994 como resultado do trabalho de uma única pessoa, Rasmus Lerdorf. O PHP foi adotado por outras pessoas inteligentes e foi reescrito três para proporcionar o amplo e aperfeiçoado produtos que é hoje. Em outubro de 2002, ele era utilizado em mais de nove milhões de domínios em todo o mundo e esse número está crescendo rapidamente. Você pode constatar o número atual em www.php.net/usage.php.

A principal versão atual do PHP é a versão 5.

Porque aprender PHP ?

Aprender PHP lhe capacitará a construir Web sites dinâmicos do mundo real, onde os usuários poderão interagir de modo significativo, diferente de um conteúdo estático de um Web site construído utilizando somente HTML.

Com o PHP é possível desenvolver diversos tipos de projetos, dentre eles podemos destacar:

- Formulários de cadastro ou contato;
- Fóruns Web;
- Sistemas de gerenciamento de conteúdo;
- Sistemas de compra e venda de produtos via Web (E-commerce);
- Sites dinâmicos alimentados pelo Banco de Dados;
- Sistemas complexos: gerenciamento de servidores, mala-direta, usuários, etc.

Até aqui você já deve ter notado as grandes possibilidades do PHP, seguindo este ponto positivo temos também a portabilidade onde o PHP está disponível para muitos sistemas operacionais diferentes. Você pode escrever o código de PHP em sistemas operacionais do tipo Unix gratuitos como Linux e o FreeBSD, versões comerciais do Unix como o Solaris e IRIX, ou em versões diferentes do Microsoft Windows. Um código bem escrito normalmente funcionará sem modificações em um sistema diferente, executando o PHP.

Neste curso abordaremos desde o básico até o avançado. Sinta-se à vontade para navegar pela apostila, voltando quando desejar.

E então, preparado? Vamos lá!

Sintaxe básica

O PHP tem uma sintaxe muito simples e enxuta, o que facilita muito a organização dos scripts a serem desenvolvidos. Outro ponto interessante que veremos é que os códigos

em PHP são embutidos no HTML, ao invés de gerá-lo por completo, facilitando muito a

análise de possíveis erros nos scripts desenvolvidos.

Delimitador e separador do código em PHP

O código de um programa escrito em PHP deve estar contido entre os seguintes delimitadores:

```
<?php
// código;
?>
```

Comentários

Existem 3 maneira de inserir comentários num código PHP. Duas delas são para comentário de uma única e linha, e a outra para comentários de mais de uma linha.

Em comentários de 1 linha, você pode usar // ou #. Já para comentários de várias linhas, delimite-o com /* e */.

```
<h1>Exemplo 1:</h1>
<?php
// Esta é uma forma de comentar 1 linha
echo 'Testando comentários no PHP.';
echo 'Teste'; // posso inserir comentário após um comando
?>

<h1>Exemplo 2:</h1>
<?php
# Esta é outra forma de comentar 1 linha
echo "Testando comentários no PHP.";
echo "Teste"; # posso inserir comentário após um comando
?>

<h1>Exemplo 3:</h1>
<?php
/* Esta forma
   Permite comentar em várias
   Linhas */
echo "Testando comentários no PHP.";
?>
```

Extensão de arquivos

A forma mais comum de nomear programas em PHP é a seguinte:

Extensão	*Significado*
.php	Arquivo PHP contendo um programa.
.class.php	Arquivo PHP contendo uma classe.
.inc.php	Arquivo PHP a ser incluído, pode incluir constantes ou configurações.

Comandos de saída(output)

Estes são comandos utilizados para gerar uma saída em tela(output).Se um script php geralmente tem como resultado uma página html, ou algum outro texto. Para gerar esse resultado, deve ser utilizada uma das funções de impressão, echo e print. Para utilizá-las deve-se utilizar um dos seguintes formatos:

```
    <?php
print ("argumento");
echo ("argumento1");
echo "argumento";
?>
```

echo

É um comando que imprime uma ou mais variáveis no console. Exemplo:

```php
<?php
echo 'a' , 'b', 'c';
?>
```

Resultado: abc

print

É uma função que imprime uma string no console. Exemplo:

```php
<?php
print ('abc');
?>
```

Resultado: abc

Tipo de Dados

O PHP suporta oito tipos primitivos.

São quatros tipos básicos:

- boolean ;
- integer ;
- float (número de ponto flutuante, ou também 'double') ;
- string .

Dois tipos compostos:

- array ;
- object .

E finalmente dois tipos especiais:

- resource ;
- NULL .

O PHP utiliza uma checagem dinâmica de tipos,isto é,uma variável pode conter valores de diferentes tipos em diferentes momentos da execução do script. Por esse motivo,não é preciso declarar o tipo de uma variável para implementá-la. O interpretador PHP decidirá qual é o tipo daquela variável,verificando o conteúdo em tempo de execução. Ainda assim,é permitido converter os valores de um tipo para outro, utilizando o *typecasting* ou a função *settype*.

Tipo Booleano

Também chamado de BOOLEAN. Este é o tipo mais fácil. Um booleano expressa um valor de verdade. Ele pode ser TRUE ou FALSE.

Para especificar um literal booleano, use as palavras chave TRUE ou FALSE. Ambas são insensitivas ao

caso. Isso quer dizer, que tanto faz você escrever “TRUE”, “True” ou “true”.

```php
<?php
$variavel = True; // assimila o valor TRUE para $variavel
?>
```

Vejamos agora exemplos sobre como fazer comparações com o tipo booleano:

```php
<?php
if ($variavel)
{
   echo "A variavel eh verdadeira";
}
else
{
   echo "A variavel eh falsa";
}

if ($variavel == TRUE)
{
   echo "A variavel eh verdadeira";
}

if (!$variavel)
{
   echo "A variavel eh falsa";
}
?>
```

No PHP pode-se fazer a conversão de tipos. Para converter explicitamente um valor para booleano, utilize-se dos modificadores (bool) ou (boolean). Entretanto, na maioria dos casos, você não precisa utilizar o modificador, desde que qualquer valor será convertido automaticamente se um operador, função ou estrutura de controle requerer um argumento booleano.Quando convertendo para booleano, os seguintes valores são considerados FALSE:

- o próprio booleano FALSE
- o inteiro 0 (zero)
- o ponto flutuante 0.0 (zero)
- uma string vazia e a string "0"
- um array sem elementos
- um objeto sem elementos membros
- o tipo especial NULL (incluindo variáveis não definidas)

Qualquer outro valor é considerado TRUE (incluindo qualquer recurso).Exemplo de conversão:

```php
<?php
$a = ""; // uma string vazia
$b = (bool) $a; // b agora é booleano e igual a FALSE

$a = 1;
$b = (bool) $a; // TRUE

$c = (bool) "string"; // TRUE
?>
```

Tipo Inteiro

Também chamado de INTEGER.Um inteiro é um número do conjunto Z = {..., -2, -1, 0, 1, 2, ...}.

Inteiros podem ser especificados em notação decimal (base 10), hexadecimal (base 16) ou octal (base 8), opcionalmente precedido de sinal (- ou +).Para usar a notação octal, você precisa preceder o número com um 0 (zero). Para utilizar a notação hexadecimal, preceda número com 0x.

Exemplo de definições:

```php
<?php
$a = 1234; # número decimal
$a = -123; # um número negativo
$a = 0123; # número octal (equivalente a 83 em decimal)
$a = 0x1A; # número hexadecimal (equivalente a 26 em decimal)
?>
```

Assim como vimos no tipo Booleano, o PHP também permite a conversão de outros tipos para Inteiro. Para converter explicitamente um valor para inteiro, utilize-se dos modificadores (int) ou (integer). Entretanto, na maioria dos casos, você não precisa utilizar o modificador, desde que qualquer valor será automaticamente convertido se um operador, função ou estrutura de controle requerer um argumento inteiro. Você também pode converter o valor de um inteiro com a função intval().

Exemplo de conversão:

```php
<?php
$a = TRUE; // booleano
$b = (int) $a; // retorna 1

$a = FALSE; // booleano
$b = (int) $a; // retorna 0

$a = 2.2232; // ponto flutuante
$b = (int) $a; // retorna 2 (convertido pra inteiro)
?>
```

Tipo Ponto Flutuante

Também chamado de FLOAT ou DOUBLE.Números de ponto flutuante podem ser especificados utilizando qualquer uma das sintaxes seguintes:

```php
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

Tipo String

String são conjunto de caracteres e pode ser definida de três formas diferentes: com apóstrofo, aspas ou sintaxe heredoc.A maneira mais simples para especificar uma string é delimitá-la entre apóstrofos (o caracter ').

Para especificar um apóstrofo dentro da string, você precisará "escapá-la" com uma contra barra (\), como em muitas outras linguagens. Se uma contra barra precisa ocorrer antes de um apóstrofo ou no final da string, você precisa duplicá-la. Note que se você tentar escapar qualquer outro caracter, a contra barra também será impressa! Então geralmente não é necessário escapar a própria contra barra.

Vejamos exemplos abaixo:

```php
<?php
echo 'isto é uma string comum';

echo 'Você pode incluir novas linhas em strings,
dessa maneira que estará
tudo bem';

echo 'Arnold once said: "I\'ll be back"';
// Imprime: Arnold disse uma vez: "I'll be back"

echo 'Você tem certeza em apagar C:\\*.*?';
// Imprime: Você tem certeza em apagar C:\*.*?

echo 'Você tem certeza em apagar C:\*.*?';
// Imprime: Você tem certeza em apagar C:\*.*?

echo 'Isto não será substituido: \n uma nova linha';
// Imprime: Isto não será substituido: \n uma nova linha

echo 'Variaveis $também não $expandem';
// Imprime: Variaveis $também não $expandem
?>
```

Em caso de dúvida,consulte a tabela seguinte:

\n	Nova linha
\r	Retorno de carro (semelhante a \n)
\t	Tabulação horizontal
\\	A própria barra (\)
\$	O símbolo $
\'	Aspa simples
\”	Aspa dupla

Tipo Array

Array, ou Vetor, é atualmente um mapa ordenado. Um mapa é um tipo que relaciona valores por chaves.Um array pode ser criado com o construtor de linguagem array(). Ele pega um certo número de pares separados por vírgula chave => valor.

Exemplo:

```php
<?php
$arr = array("chave" => "valor", "chave2" => "valor2", 3 => "valor3");

echo $arr["chave"];  // valor
echo $arr["chave2"]; // valor2
echo $arr[3];        // valor3
?>
```

Observe que a chave tanto pode ser uma string (“chave”) quanto um inteiro (3).Já o valor pode conter qualquer tipo suportado pelo PHP. Um valor pode ainda, conter outro array, como segue no exemplo:

```php
<?php
$arr = array("chave" => array(6 => 5, 13 => 9, "a" => 42));

echo $arr["chave"][6];     // 5
echo $arr["chave"][13];    // 9
echo $arr["chave"]["a"];   // 42
?>
```

Além de definir arrays com o comando array(), você ainda pode definir diretamente com atribuição. Veja o exemplo:

```php
<?php
$arr[0] = 'Valor x';
$arr[1] = 'Valor y';

echo $arr[0]; // Valor x
echo $arr[1]; // Valor y
?>
```

Como vimos, as chaves podem conter inteiros. Você pode ainda inserir novos valores num array, apenas usando os colchetes. Continuando o exemplo anterior:

```php
<?php
$arr[0] = 'joão';  // chave 0
$arr[1] = 'maria'; // chave 1
$arr[] = 'josé';   // chave 2
$arr[] = 'paulo';  // chave 3

echo $arr[0]; // joão
echo $arr[1]; // maria
echo $arr[2]; // josé
echo $arr[3]; // paulo
?>
```

Para apagar uma chave do vetor, você pode utilizar a função unset(). Como no exemplo a seguir:

```php
<?php
$arr[0] = 'joão';  // chave 0
$arr[1] = 'maria'; // chave 1

unset($arr[0]);

echo $arr[0]; // não existe a chave 0, portanto não imprime
?>
```

Mais informações sobre Arrays, você irá visualizar na seção dedicada: Trabalhando com Arrays.

Tipo Objetos

Objetos são instâncias de classes definidas pelo usuário.Para inicializar um objeto, você usa a instrução new, criando uma instância do objeto em uma variável.

Exemplo:

```php
<?php
class foo
{
    function do_foo()
    {
        echo "Fazendo foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

Tipo NULL

O valor especial NULL representa que a variável não tem valor. NULL é o único valor possível do tipo NULL.Há apenas um único valor do tipo NULL, e é a palavra (insensitiva ao caso) NULL.

Exemplo:

```php
<?php
$var = NULL;
?>
```

Para verificar se uma variável é NULL, utilize a função is_null(), passando como parâmetro a variável.
Exemplo:

```php
<?php
$var = NULL;
if (is_null($var))
{
   echo "A variável foi setada como NULL";
}
?>
```

Variáveis

As variáveis no PHP são representadas por um cifrão ($) seguido pelo nome da variável. Os nomes de variável no PHP fazem distinção entre maiúsculas e minúsculas.

Os nomes de variável seguem as mesmas regras como outros rótulos no PHP. Um nome de variável válido se inicia com uma letra ou sublinhado, seguido de qualquer número de letras, algarismos ou sublinhados. Não é válido, portanto, variáveis que iniciem com números.

Exemplo:

```php
<?php
$var = "Grupo";
$Var = "E-jovem";
echo "$var, $Var";      // exibe "Grupo, E-jovem"

$2var = 'teste';      // inválido; começa com um número
$_2var = 'teste';     // válido; começa com um sublinhado
?>
```

Constantes

Uma constante é um identificador (nome) para um único valor. Como o nome sugere, esse valor não pode mudar durante a execução do script. As constantes são sensíveis ao caso por padrão. Por convenção, os nomes de constantes são sempre em maiúsculas.

O nome de uma constante tem as mesmas regras de qualquer identificador no PHP. Um nome de constante válida começa com uma letra ou sublinhado, seguido por qualquer número de letras, números ou sublinhados.Você pode definir uma constante utilizando-se da função define(). Quando uma constante é definida, ela não pode ser mais modificada ou anulada.

Estas são as diferenças entre constantes e variáveis:

- Constantes não podem ter um sinal de cifrão ($) antes delas;
- Constantes só podem ser definidas utilizando a função define(), e não por simples assimilação;
- Constantes podem ser definidas e acessadas de qualquer lugar sem que as regras de escopo de variáveis seja aplicadas;
- Constantes não podem ser redefinidas ou eliminadas depois que elas são criadas;
- Constantes só podem conter valores escalares.

Os tipos de dados aceitos numa constante são: Booleano, Inteiro, Float e String.Exemplo de definição de constante:

```php
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // imprime "Hello world."
echo Constant; // imprime "Constant" e gera um alerta notice.
?>
```

Operadores

Um operador é algo que você alimenta com um ou mais valores (ou expressões, no jargão de programação) e que devolve outro valor (e por isso os próprios construtores se tormam expressões). Assim, você pode pensar que as funções e os construtores que retornam valores (como o print) são operadores e os outros que não retornam nada (como echo) como uma outra coisa.

Há três tipos de operadores. Primeiramente, os operadores unários, que operam em apenas um valor. Por exemplo, ! (operador de negação) ou o ++ (operador de incremento). No segundo grupo estão os operadores binários, o o grupo que contém a maioria dos operadores que o PHP suporta, com uma lista completa logo abaixo.

O terceiro grupo é do operador ternário: ?:. Ele pode ser usado para selecionar entre dois valores dependendo de uma terceira, em vez de selecionar duas sentenças ou encadeamentos de execução. Englobar expressões ternárias com parênteses é uma boa idéia.

Operadores de atribuição

O operador básico de atribuição é "=".

A sua primeira vista deve ser a de definir isto como "é igual". Não. Isto quer dizer, na verdade, que o operando da esquerda recebe o valor da expressão da direita, ou seja, "é configurado para".

O valor de uma expressão de atribuição é o valor atribuído. Ou seja, o valor de "$a = 3" é 3. Isto permite que você faça alguns truques.

No exemplo abaixo, $a é igual a 9 agora e $b foi configurado como 4:

```php
<?php
$a = ($b = 4) + 5;
?>
```

Há ainda o que chamamos de operadores combinados. Significa combinar operadores aritméticos e de concatenação com o operador de atribuição.

Exemplo: vamos verificar como concatenar duas strings, fazendo uso dos operadores combinados.

```php
<?php
$a = 'Bom';
$a = 'Dia!';
echo $a; // imprime "Bom Dia!" na tela
?>
```

Acima o operador ".=" significa concatenar com o valor anterior da variável, o novo valor passado.

```php
<?php
$a = 5;
$a += 3;
echo $a; // Imprime "8" na tela
?>
```

Vejamos outro exemplo, com operador aritmético:

O operador “+=” soma à variável, o valor passado. Neste caso foi somado o valor 3 ao valor 5. Da mesma forma você pode usar “-=” para subtração, “*=” para multiplicação, etc.

Operadores aritméticos

Lembra-se da aritmética básica da escola? Estes operadores funcionam exatamente como aqueles.

Exemplo	*Nome*	*Resultado*
$a + $b	Adição	Soma de $a e $b.
$a - $b	Subtração	Diferença entre $a e $b.
$a * $b	Multiplicação	Produto de $a e $b.
$a / $b	Divisão	Quociente de $a por $b.
$a % $b	Módulo	Resto de $a dividido por $b.

O operador de divisão ("/") sempre retorna um valor com ponto flutuante (fracionário), mesmo que os dois operandos sejam inteiros (ou strings convertidos para inteiros).

Operadores de comparação

Operadores de comparação, como os seus nomes implicam, permitem que você compare dois valores. A tabela abaixo lista os operadores de comparação existentes no PHP:

Exemplo	*Nome*	*Resultado*
$a == $b	Igual	Verdadeiro (**TRUE**) se $a é igual a $b.
$a === $b	Idêntico	Verdadeiro (**TRUE**) se $a é igual a $b, e eles são do mesmo tipo (somente para PHP4).
$a != $b	Diferente	Verdadeiro se $a não é igual a $b.
$a <> $b	Diferente	Verdadeiro se $a não é igual a $b.
$a !== $b	Não idêntico	Verdadeiro de $a não é igual a $b, ou eles não são do mesmo tipo (somente para o PHP4).
$a < $b	Menor que	Verdadeiro se $a é estritamente menor que $b.
$a > $b	Maior que	Verdadeiro se $a é estritamente maior que $b.
$a <= $b	Menor ou igual	Verdadeiro se $a é menor ou igual a $b.
$a >= $b	Maior ou igual	Verdadeiro se $a é maior ou igual a $b.

Outro operador condicional é o operador "?:" (ou ternário), que opera como no C e em muitas outras linguagens. Vejamos um exemplo do operador “?:”:

```php
<?php
$a = true;
$b = ($a) ? 'Verdadeiro' : 'Falso';
echo $b; // imprime "Verdadeiro" na tela
?>
```

Explicando: o operador “?:” funciona como um IF e ELSE, mas numa mesma linha, para operações simples condicionais. Neste caso, se a variável $a for verdadeira (TRUE) então a variável $b assume a string “Verdadeiro”, caso contrário, a variável $b assume a string “Falso”.Da mesma forma, poderíamos fazer:

```php
<?php
$a = true;
if ($a)
{
    $b = "Verdadeiro";
}
else
{
    $b = "Falso";
}
echo $b;
?>
```

Operadores lógicos

Os operadores lógicos, assim como os de comparação, examina 2 variáveis e retorna um resultado binário Falso (FALSE) ou Verdadeiro (TRUE).

A tabela abaixo lista os tipos de operadores lógicos:

Exemplo	*Nome*	*Resultado*
$a and $b	E	Verdadeiro (**TRUE**) se tanto $a quanto $b são verdadeiros.
$a or $b	OU	Verdadeiro se $a ou $b são verdadeiros.
$a xor $b	XOR	Verdadeiro se $a ou $b são verdadeiros, mas não ambos.
! $a	NÃO	Verdadeiro se $a não é verdadeiro.
$a && $b	E	Verdadeiro se tanto $a quanto $b são verdadeiros.
$a || $b	OU	Verdadeiro se $a ou $b são verdadeiros.

A razão para as duas variantes dos operandos "and" e "or” é que eles operam com precedências diferentes.

Operadores de strings

Há dois operadores de string.

O primeiro é o operador de concatenação “.”, que retorna a concatenação dos seus argumentos direito e esquerdo.

O segundo é o operador de atribuição de concatenação “.=” (previamente explicado em Operadores de atribuição), que acrescenta o argumento do lado direito no argumento do lado esquerdo.

Exemplo:

```php
<?php
$a = "Olá ";
$b = $a . "mundo!"; // agora $b contém "Olá mundo!"

$a = "Olá ";
$a .= "mundo!";     // agora $a contém "Olá mundo!"
?>
```

Precedência de Operadores

A precedência de um operador especifica quem tem mais prioridade quando há duas delas juntas.

Por exemplo, na expressão, 1 + 5 * 3, a resposta é 16 e não 18 porque o operador de multiplicação ("*") tem prioridade de precedência que o operador de adição ("+").

Parênteses podem ser utilizados para forçar a precedência, se necessário. Assim, (1 + 5) * 3 é avaliado como 18.

Expressões

Expressões são as peças de construção mais importantes do PHP. No PHP, quase tudo o que você escreve são expressões. A maneira mais simples e ainda mais precisa de definir uma expressão é "tudo o que tem um valor". As formas mais básicas de expressões são constantes e variáveis. Quando você digita "$a = 5", você está atribuindo '5' para $a. '5', obviamente, tem o valor 5 (neste caso, '5' é uma constante inteira). Depois desta atribuição, você pode esperar que o valor de $a seja 5. Assim se você escrever $b = $a, você pode esperar que $b se comporte da mesma forma que se você escrevesse $b = 5. Há mais uma expressão que podem parecer estranha se você não a viu em outras linguagens, o operador condicional ternário:

```php
<?php
$primeira ? $segunda : $terceira
?>
```

Explicando: se o valor da primeira sub-expressão é verdadeiro (TRUE, não-zero), então a segunda sub-expressão é avaliada, e este é o resultado da expressão condicional. Caso contrário, a terceira sub-expressão é avaliada e este é o valor. Poderia ser escrita também assim:

```php
<?php
if ($primeira)
{
      $segunda;
}
else
{
      $terceira;
}
?>
```

Outro tipo comum de expressão, são as expressões de comparação. O PHP suporta:

Simbolo	*Significado*
>	Maior que
>=	Maior ou igual
==	Igual

!=	diferente
<	menor que
<=	menor ou igual
===	igual e do mesmo tipo
!==	diferente ou não do mesmo tipo

Estas expressões são usadas mais freqüentemente dentro de instruções condicionais, como em comandos if.

Outro bom exemplo de orientação de expressão é o pré e o pós-incremento e decremento. Se você quer somar 1 à variável $a, pode fazer de 2 formas:

```
<?php
$a = $a + 1; // soma 1 à variavel $a
$a++; // pós-incremento = também soma 1 à variável $a
++$a; // pré-incremento = também soma 1 à variável $a
?>
```

Da mesma forma você pode usar para subtração. A diferença entre pré e pós-incremento, é de que no pré-incremento o valor é calculado antes e retornado depois. No pós-incremento, o valor é retornado, e só depois calculado.Ainda em se tratando de expressões, outro caso importante é o de combinar operador-atribuição.

Você já sabe que se você quer incrementar $a de 1, você só precisa escrever $a++ ou ++$a. Mas e se você quiser somar mais que um a ele, por exemplo 3? Você poderia escrever $a++ várias vezes, mas esta obviamente não é uma forma muito eficiente ou confortável.

Uma prática muito mais comum é escrever $a = $a + 3. $a + 3 é avaliada como o valor de $a mais 3, e é atribuído de volta a $a, que resulta em incrementar $a de 3. Em PHP, como em várias outras linguagens como o C, você pode escrever isto de uma forma mais curta, que com o tempo se torna mais limpa e rápida de se entender também. Somar 3 ao valor corrente de $a pode ser escrito $a +=3. Qualquer operador de dois parâmetros pode ser usado neste modo operador-atribuição, por exemplo $a -= 5 (subtrai 5 do valor de $a), $ b *= 7 (multiplica o valor de $b por 7), etc.O exemplo a seguir mostra os tipos de expressões explicados nesta parte:

```php
<?php
function double($i)
{
    return $i*2;
}
/* atribui o valor cinco às variáveis $a e $b */
$b = $a = 5;
/* pós-incremento, atribui o valor original
   de $a (5) para $c, e depois incrementa $a (6) */
$c = $a++;
/* pré-incremento, atribui o valor incrementado
   de $b (6) a $d e $e */
$e = $d = ++$b;

/* neste ponto, tanto $d quanto $e são iguais a 6 */

/* atribui o dobro do valor de $d antes
   do incremento, 2*6 = 12 a $f */
$f = double($d++);
/* atribui o dobro do valor de $e depois
   do incremento, 2*7 = 14 a $g */
$g = double(++$e);
/* primeiro, $g é incrementado de 10 e termina
   com o valor 24. O valor da atribuição (24) é
   então atribuído a $h, e $h termina com o valor
   24 também. */
$h = $g += 10;
?>
```

Trabalhando com Arrays

Já vimos no capítulo sobre Tipos, que um Array, ou Vetor, é um mapa ordenado que relaciona valores por chaves. Neste capítulo vamos aprofundar um pouco mais os estudos neste tipo de dados, que é largamente utilizados em muitas operações no PHP.

O que é um array?

Uma variável escalar é uma localização identificada com um nome em que é armazenado um valor; de maneira semelhante, um array é um lugar identificado com um nome para armazenar um conjunto de valores, permitindo que você agrupe valores escalares.

Uma lista de frutas será o array para nosso exemplo. Na figura abaixo, você pode ver uma lista de três frutas armazenadas em um formato de array e uma variável, chamada $cesta, que armazena os três valores. (Veremos como criar uma variável como essa em um minuto).

Os valores armazenados em um array são chamados elementos do array. Cada elemento do array tem um índice associado (também denominado chave) que é utilizado para acessar o elemento. Os arrays, na maioria das linguagens de programação, têm índices numéricos que geralmente iniciam em zero ou um.

O PHP permite o uso de números e strings como índices de array. É possível usar arrays da maneira tradicional indexada numericamente ou agrupar as chaves para que a indexação seja mais significativa e útil. (Essa abordagem deve ser familiar caso você já tenha usado arrays associativos ou mapas em outras

linguagens de programação.) A abordagem de programação pode variar um pouco dependendo se você está usando array indexados numericamente de forma padrão ou valores de índice mais interessantes.

Iniciaremos examinando arrays numericamente indexados e depois veremos as chaves definidas pelo usuário.

Arrays numericamente indexados

Esses arrays são suportados na maioria das linguagens de programação. Em PHP, os índices iniciam no zero por padrão, embora você possa alterar isso.

Inicializando arrays numericamente indexadosPara criar o array mostrado na figura logo acima, em nossa

```
$cesta = array("Pêra", "Uva", "Maçã");
```

cesta de frutas, utilize a seguinte linha de código de PHP:

Isso criará um array chamado $cesta contendo os três valores dados – “Pêra”, “Uva” e “Maçã”. Note que, como echo, array( ) é na realidade uma construção da linguagem em vez de uma função.

Dependendo do conteúdo que você precise no array, talvez não seja necessário inicializá-lo manualmente como no exemplo anterior. Se tiver os dados de que precisa em outro array, você simplesmente pode copiar um array para outro utilizando o operador =.

Acessando o conteúdo de array

Para acessar o conteúdo de uma variável, utilize o nome dela. Se a variável for um array, acesse o conteúdo utilizando o nome da variável e uma chave ou índice. A chave ou índice indica quais valores armazenados acessamos. O índice é colocado entre colchetes depois do nome.

Digite $cesta[0], $cesta[1], $cesta[2] para utilizar o conteúdo do array $cesta. Por padrão, o elemento zero é o primeiro elemento do array. Esse é o mesmo esquema de numeração utilizado em C, C++, Java e em várias outras linguagens, mas talvez você precise de algum tempo para se acostumar com ele se não conhecê-lo.

Como com outras variáveis, o conteúdo dos elementos do array é alterado utilizando o operador =. A

```
$cesta[0] = "Banana";
```

próxima linha substituirá o primeiro elemento no array “Pêra” por “Banana”.

A linha a seguir poderia ser utilizada para adicionar um novo elemento – “Banana” – ao final do array,

```
$cesta[3] = "Banana";
```

fornecendo um total de quatro elementos:

```
echo "$cesta[0] $cesta[1] $cesta[2] $cesta[3]";
```

Para exibir o conteúdo, poderíamos digitar:

Observe que embora a análise sintática da string de PHP seja relativamente inteligente, você pode confundi-la. Se você estiver tendo problemas com arrays ou outras variáveis que não são interpretadas

corretamente quando são inseridas dentro de uma string entre aspas duplas, pode colocá-las fora das aspas. A instrução echo anterior funcionará corretamente, mas, em muitos dos exemplos mais complexos, você notará que as variáveis está fora das strings entre aspas.

Como com outras variáveis de PHP, os arrays não precisam ser inicializados ou criados antecipadamente. Eles são criados de forma automática na primeira vez em que você os utiliza.

```
$cesta[0] = "Pêra";
$cesta[1] = "Uva";
$cesta[2] = "Maçã";
```

O código a seguir criará o mesmo array $cesta criado anteriormente com a instrução array( ):

Se $cesta não existir ainda, a primeira linha criará um novo array com apenas um elemento. As linhas subseqüentes adicionam valores ao array. O array é redimensionado dinamicamente à medida que você adiciona elementos a ele. Essa funcionalidade de redimensionamento não está presente na maioria das outras linguagens de programação.

Utilizando loops para acessar o array

Como o array está indexado por uma seqüência de números, podemos utilizar um loop for mais facilmente para exibir o conteúdo:

```
for ( $i = 0; $i < 3; $i++ )
    echo "$cesta[$i] ";
```

Esse loop fornecerá saída semelhante ao código anterior, mas exigirá menos digitação do que escrever o código manualmente para trabalhar com cada elemento em um array grande. A capacidade de utilizar um loop simples para acessar cada elemento é um excelente recurso de arrays. Também podemos usar o loop foreach, visto anteriormente, especialmente projetado para usar com os arrays, poderíamos utilizá-lo como a seguir:

```
foreach ($cesta as $chave=>$valor)
    echo "$chave : $valor ";
```

Por sua vez, esse código armazena cada índice do array em $chave e cada valor em $valor e os imprime.

Arrays Associativos

No array $cesta, permitimos que o PHP ofereça um índice padrão a cada item. Isso significa que o primeiro item que adicionamos tornou-se o item 0, o segundo item, 1 e assim por diante. O PHP também suporta arrays em que podemos associar qualquer chave ou índice que quisermos a cada valor.

Inicializando um array

O código a seguir cria um array associativo com nome de frutas como chaves, e preços como valores.

```
$precos = array( "Pêra"=>7, "Uva"=>3, "Maçã"=>1 );
```

O símbolo entre as chaves e os valores é simplesmente um sinal de igual seguido imediatamente de um sinal de maior que.

Acessando os elementos do array

Novamente, acessamos o conteúdo utilizando o nome da variável e uma chave, então podemos acessar as informações que armazenamos no array de preços como $precos[“Pêra”], $precos[“Uva”] e $precos[“Maçã”].

O código a seguir criará o mesmo array $precos. Em vez de criar um array com três elementos, essa

```
$precos = array( "Pêra"=>7 );
$precos["Uva"] = 3;
$precos["Maçã"] = 1;
```

versão cria um array com somente um elemento e então acrescenta outros dois.

Eis outra parte, ligeiramente diferente, mas equivalente de código. Nesta versão, não criamos explicitamente um array. O array é automaticamente criado quando adicionamos a ele o primeiro

```
$precos["Pêra"] = 7;
$precos["Uva"] = 3;
$precos["Maçã"] = 1;
```

elemento.

Arrays multidimensionais

Os arrays não têm de ser uma lista simples de chaves e valores – cada localização no array pode armazenar outro array. Dessa maneira, podemos criar um array bidimensional. Você pode pensar em um array de duas dimensões como sendo uma matriz ou grade, com largura e altura ou linhas e colunas.

Se quiséssemos armazenar mais de uma parte de dados sobre cada item da cesta, poderíamos utilizar um array bidimensional.

A figura abaixo mostra os itens da cesta representados como um array bidimensional com cada linha representando um produto individual e cada coluna representando um atributo do produto armazenado.

Utilizando PHP, escreveríamos o código a seguir para configurar os dados no array mostrado na figura acima.

```
$cesta = array( array( "PER", "Pêra", 7 ),
                array( "UVA", "Uva", 3 ),
                array( "MAC", "Maçã", 1 ) );
```

Você pode ver a partir dessa definição que nosso array de produtos agora contém três arrays. Para acessar os dados em um array dimensional, lembre-se de que precisamos do nome dos arrays e do índice dos elementos. Um array bidimensional é semelhante, exceto que cada elemento tem dois índices – uma linha e uma coluna. (A linha superior é a linha 0 e a coluna na extremidade esquerda é a coluna 0.)

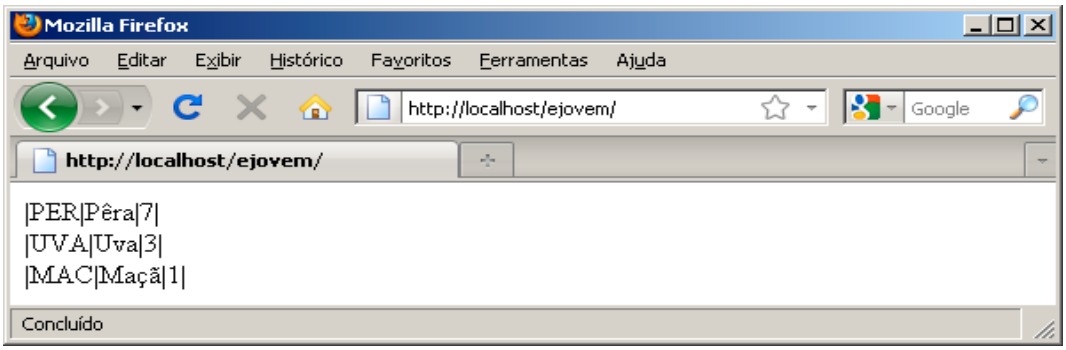
Para exibir o conteúdo desse array, poderíamos acessar manualmente cada elemento no pedido deste modo:

```
echo '|'.$cesta[0][0].'|'.$cesta[0][1].'|'.$cesta[0][2].'|<br/>';
echo '|'.$cesta[1][0].'|'.$cesta[1][1].'|'.$cesta[1][2].'|<br/>';
echo '|'.$cesta[2][0].'|'.$cesta[2][1].'|'.$cesta[2][2].'|<br/>';
```

```
for ( $linha = 0; $linha < 3; $linha++ )
{
    for ( $coluna = 0; $coluna < 3; $coluna++ )
    {
        echo '|'.$cesta[$linha][$coluna];
    }
    echo '|<br/>';
}
```

Alternativamente, poderíamos colocar um loop for dentro de outro para alcançar o mesmo resultado.

As duas versões desse código produzem a mesma saída no navegador:



Estruturas de Controle

As estruturas que veremos a seguir são comuns para as linguagens de programação imperativas, bastando, portanto, descrever a sintaxe de cada uma delas, resumindo o funcionamento.

IF

O IF é usado como estrutura condicional. Indica que se uma expressão for verdade, então faça tal ação.Sua sintaxe é simples. Se a ação a ser executada for de uma única linha, as chaves são opcionais, caso contrário, se a ação tiver vários comandos a serem executados, as chaves são obrigatórias.

Veja um exemplo:

```php
<?php
$a = 3;
$b = 1;

if ($a > b)
    echo "a é maior que b"; // único comando

if ($a > b)
{
   // varios comandos, uso obrigatorios das chaves
   $b = a;
   $a++;
}
?>
```

ELSE

O ELSE é usado para que uma ação seja executada caso a expressão analisada no IF seja falsa. Em outras palavras, se uma expressão for verdade, faça a ação do IF, senão faça a ação do ELSE.

Veja um exemplo:

```php
<?php
$a = 1;
$b = 5;

if ($a > b)
{
   echo "a é maior que b";
}
else
{
   echo "b é maior que a";
}
?>
```

ELSEIF

A estrutura ELSEIF faz com que o usuário possa definir várias situações para que uma ação ocorra. Suponha que você quer executar uma ação para cada valor da uma variável. Por exemplo: se $var for igual a x, execute tal ação, se for igual a y, outra ação, se for igual a z, outra ação, e assim por diante.

Exemplo:

```php
<?php
$var = 'y';

if ($var == 'x')
{
     echo "Variavel eh X";
}
elseif ($var == 'y')
{
     echo "Variavel eh Y";
}
elseif ($var == 'z')
{
     echo "Variavel eh Z";
}
else
{
     echo "Variavel tem outro valor";
}
?>
```

SWITCH

A instrução switch é similar a uma série de instruções IFs seguidas.

Os exemplos seguintes mostram duas maneiras diferentes de escrever a mesma coisa, uma utilizando uma série de IFs e ELSEIFs e a outra utilizando a instrução switch:

```php
<?php
if ($i == 0)
{
    echo "i igual a 0";
}
elseif ($i == 1)
{
    echo "i igual a 1";
}
elseif ($i == 2)
{
    echo "i igual a 2";
}
switch ($i)
{
    case 0:
        echo "i igual a 0";
        break;
    case 1:
        echo "i igual a 1";
        break;
    case 2:
        echo "i igual a 2";
        break;
}
?>
```

É importante entender como a instrução switch funciona para evitar enganos.

A instrução switch executa linha a linha. No início, nenhum código é executado. Somente quando uma instrução case é encontrada com um valor que combina com a expressão do switch faz com que o PHP execute as instruções a partir daí.O PHP continua executando as instruções até o fim do bloco switch ou na primeira vez que encontrar uma instrução break. Se você não escrever uma instrução break no fim das instruções case, o PHP continuará executando os cases seguintes.

Um case especial é o default. Esse case é executado quando nenhum outro case combina. Ele precisa ser a última instrução case. Por exemplo:

```php
<?php
switch ($i)
{
    case 0:
        echo "i igual a 0";
        break;
    case 1:
        echo "i igual a 1";
        break;
    case 2:
        echo "i igual a 2";
        break;
    default:
        echo "i não é igual a 0, 1 ou 2";
}
?>
```

BREAK

BREAK cancela a execução do comando for, foreach while, do..while ou switch atual.Break aceita um argumento numérico opcional que diz a ele quantas estruturas aninhadas englobadas devem ser quebradas.

CONTINUE

CONTINUE é usado dentro de estruturas de loops para saltar o resto da iteração do loop atual e continuar a execução no início da próxima iteração.Continue aceita um argumento numérico opcional que diz a ele de quantos níveis de loops aninhados ele deve saltar até o fim.

```
<?php
$i = 0;
while ($i++ < 5) {
echo "Fora<br>\n";
while (1) {
echo "  Meio<br>\n";
while (1) {
echo "  Dentro<br>\n";
continue 3;
}
echo "Isto nunca será exibido.<br>\n";
}
echo "Nem isso.<br>\n";
}
?>
```

Ilustração : ex_continue.php

FOR

A estrutura de laço (loop) FOR, realiza repetidas ações a depender das expressões passadas como parâmetro. No total são 3 expressões:

```
for (expr1; expr2; expr3)
{
   // ações
}
```

A primeira expressão (expr1) é avaliada (executada) uma vez no começo do loop. No começo de cada iteração, expr2 é avaliada. Se ela é avaliada como TRUE, o loop continua e o(s) comando(s) aninhado(s) é(são) executado(s). Se é avaliada como FALSE, a execução do 'loop' termina. No fim de cada iteração, expr3 é avaliada (executada).

Veja um exemplo que imprime números de 1 a 10 na tela:

```
<?php
for ($i = 1; $i <= 10; $i++)
{
    echo $i;
}
?>
```

Explicando: a variável $i é iniciada com valor 1, em seguida é feita a verificação ($i <= 10) se for verdade entra no loop, caso contrário termina, e em cada iteração a variável $i é incrementada em 1 ($i++).

FOREACH

O construtor foreach permite interagir com arrays de forma mais simples, em força de laço, percorrendo todo o vetor.É útil caso você queira, por exemplo, usar os valores de todo o vetor para realizar alguma operação, ou imprimir na tela.

Tem-se 2 tipos de sintaxe. A primeira não leva em consideração a chave do array, e a segunda utiliza as chaves.

```php
<?php
foreach (expressao_array as $valor) //instrucoes
foreach (expressao_array as $chave => $valor)// instrucoes
?>
```

Sintaxe:

Vejamos um exemplo:

```php
<?php
// exemplo usando apenas valores
$a = array(1, 2, 3, 17);
 foreach ($a as $v) {
   echo "Valor atual: $v.\n";}
// exemplo usando chaves e valores
$a = array (
    "um" => 1,
    "dois" => 2,
    "três" => 3,
    "dezessete" => 17 );
 foreach ($a as $k => $v) {
    echo "\$a[$k] => $v.\n"; }
 ?>
```

WHILE

While é outra estrutura de laço (loop) mais simples. Enquanto a expressão (passada como parâmetro) for verdade, será executada ações dentro do bloco de chaves.

```
while (expressao)
{
   // ações
}
```

Vejamos o mesmo exemplo de impressão de números de 1 a 10, mas agora utilizando a estrutura While:

```php
<?php
$i = 1;
while ($i <= 10)
{
    echo $i;
    $i++; // incrementa o i para proxima iteracao
}
?>
```

Observe que o incremento é feito manualmente dentro do laço. Se você esquecer de incrementar, neste exemplo, o script ficará em loop infinito e será abortado pelo PHP.

DO...WHILE

O laço do...while funciona de maneira bastante semelhante ao while, com a simples diferença que a expressão é testada ao final do bloco de comandos.Isso significa que a ação é feita no mínimo 1 vez.

O laço do...while possui apenas uma sintaxe, que é a seguinte:

```php
<?php
$i = 0;

do
{
    echo $i;
    $i++; // incrementa o i para proxima interacao
}
while ($i < 10);
?>
```

Funções

Uma função é um pedaço de codigo com um objetivo especifico,encapsulado sob uma estrutura unica que recebe um conjunto de parâmetros e retorna um dado.Uma função é declarada uma unica vez ,mas pode ser utilizada diversas vezes. É uma das estruturas mais básicas para prover reusabilidade.

Criação

Para declarar uma função em PHP,utiliza-se o operador function seguido do nome que desejamos lhe atribuir ,sem espaços em branco e iniciando obrigatoriamente com uma letra. Na mesma linha ,digitamos a lista de argumentos (parâmetros) que a função irá receber,separados por virgula. Em seguida ,encapsulado por chaves {}, vem o código da função. No final ,utiliza-se a clausula return para retornar o resultado da função (integer,strig,array,objeto etc).

```php
<?php
//exemplo de função
function nome_de_funcao ($arg1, $arg2, $argN)
{
$valor =  $arg1 + $arg2 + $argN;
return $valor;
}
?>
```

No exemplo a seguir criamos uma função que calcula o índice de obesidade de alguem. A função recebe dois parâmetros ($peso e $altura) e retorna um valor definido por uma formula.

```php
<?
function calcula_obesidade($peso, $altura)
{
return $peso / ($altura * $altura);
}
echo calcula_obesidade(70, 1.85);
?>
```

Resultado: 20.45288531775

Retornando Valores de uma Função

É possível a uma função retornar um valor,o qual pode ser de qualquer tipo aceito pelo PHP (inteiro,ponto flutuante,booleano,string,array,etc.).

Para isso devemos utilizar a instrução return seguida do valor que desejamos retornar 9ou expressão).

```php
<?php
function cubo($valor) {
$cubo = $valor*$valor*$valor;
return $cubo;
}
$valor = 7;
echo "O cubo de $valor é " . cubo($valor);
?>
```

O resultado desse script é: O cubo de 7 é 343

Valores são retornados pelo uso de comandos opcionais de retorno. Qualquer tipo pode ser retornado, incluindo listas e objetos.

```php
<?php
function teste ($numero) {
return $numero+$numero*2;
}
echo teste (7); // imprime '21'.
?>
```

Ilustração : func_return_val.php

Criando bloco de códigos reutilizáveis

Quando estiver projetando seu site da Web, você apontará alguns elementos encontrados em muitas páginas em todo o site. Tais elementos podem ser barras de navegação ou uma linha de base com o endereço de e-mail do seu webmaster. Outros elementos podem ser fragmentos de códigos para exibir o dia, dados computados financeiros padrão e muitos outros códigos padronizados de HTML ou PHP, como por exemplo, constantes.

require( )

A instrução **require( )** e **include( )** são idênticos em todas as formas exceto pela manipulação de erros.O **include( )** produz **Warning** enquanto **require( )** produzirá um **Fatal Error**. Em outras palavras, não hesite em utilizar **require( )** se na falta de um arquivo quiser parar o processamento da página.O **include( )** não se comporta da mesma maneira, e o script poderá continuar nessa situação.

Veja os exemplos abaixo de sua utilização:

```php
<?php
require 'teste.php';
require $arquivo;
require ('arquivo.txt');
?>
```

Nota: Até o PHP 4.0.2, havia o seguinte comportamento: **require( )** ocorre mesmo que a linha onde ele está nunca seja executada. É por isso que instruções condicionais não afetam **require( )**. Entretanto, se a linha onde ocorre o **require( )** não for executada, nada do código incluído do arquivo também será. Similarmente, estruturas de loop não afetam o funcionamento do **require( )**. Mas o código incluído pela função será *submetido* ao loop. A instrução **require( )** apenas ocorre uma vez.

include( )

A instrução **include( )** inclui e avalia o arquivo informado. Sua semelhança com o require dispensa maiores explicações.Qualquer variável disponível da linha onde a chamada da inclusão ocorre estará disponível para o arquivo incluído, daquele ponto em diante.

Veja os exemplos de include( ):

```php
<?php
$nome = 'Leonardo';
$apelido = 'Leo';
?>
```

Ilustração : includ.php

```php
<?php
echo "O nome é $nome e seu apelido é $apelido";
//As variáveis estão vazias
include 'includ.php';
echo " O nome é $nome e seu apelido é $apelido";
// As variáveis neste caso contém as informações inclusas
?>
```

Ilustração : ex_include.php

Se o include ocorre dentro de uma função do arquivo principal, então todo o código incluído será executado como se ele tivesse sido definido dentro daquela função. Da mesma forma, ele seguirá o escopo de variáveis da função.

```php
<?php
function teste()
{
global $nome;
include 'includ.php';
echo " O nome é $nome e seu apelido é $apelido";
}
/* includ.php está no escopo da função teste( ),então $apelido
NÃO está disponível fora de seu escopo. $nome estará porque ela
foi declarada como global */
teste( );// Imprime o conteúdo da variável $nome e da variável $apelido
echo "O nome é $nome e o apelido é $apelido"; //Imprime somente $nome
?>
```

Ilustração : ex_include_2.php

Quando um arquivo é incluído, o interpretador sai do modo PHP e entra no modo HTML (no começo do arquivo incluído), e alterna novamente no seu fim. Por isso,qualquer código dentro do arquivo incluído que precisa ser executado como código PHP tem de ser delimitado por tags lidas de abertura e fechamento.Se "**URL fopen wrappers**" estão ativas no PHP (normalmente na configuração padrão), você pode especificar um arquivo utilizando uma URL (via HTTP) em vez de um caminho local. Se o servidor apontado interpreta o arquivo informado como código PHP, variáveis podem ser passadas ao arquivo incluído na URL de requisição como num HTTP GET. Isto não é necessariamente a mesma coisa que incluir o arquivo e compartilhar o escopo de variável do arquivo principal: o script será executado no servidor remoto e apenas seu resultado será incluído no script local.

```php
<?php
/* Este exemplo assume que www.exemplo.com está configurado para
interpretar arquivos .php mas não .txt. Além, 'Funciona' aqui
significa que as variáveis $teste e $teste2 estão disponíveis no
arquivo incluído*/
/*Não funciona: arquivos .txt não são manipulados em
www.exemplo.com como PHP */
include 'http://www.exemplo.com/arquivo.txt?teste=1&teste2=2';
/* Não funciona: procura por um arquivo chamado
arquivo.php?teste=1&teste2=2' no sistemas de arquivo local. */
include 'arquivo.php?teste=1&teste2=2';
// Funciona.
include 'http://www.exemplo.com/arquivo.php?teste=1&teste2=2';
$teste = 1;
$teste2 = 2;
include 'arquivo.txt'; // Funciona.
include 'arquivo.php'; // Funciona.
?>
```

Por serem **include( )** e **require( )** dois *construtores de linguagem especiais*, você precisa delimitá-los como um bloco de instruções quando utilizados dentro de instruções condicionais.

```php
<?php
// Isto está errado e não funcionará como desejado
if ($condicao)
include $arquivo;
else
include $outro;
// E este está correto
if ($condicao) {
include $arquivo;
} else {
include $outro;
}
?>
```

Também é possível executar uma instrução **return( )** dentro de um arquivo incluído de maneira a finalizar o processamento daquele arquivo e retornar para o script que o chamou. Também é possível retornar valores de arquivos incluídos. Você pode pegar o valor de retorno de um include como faria com uma função normal. O **return( )** se aplica apenas para a função e não para todo o arquivo.

```php
<?php
$var = 'PHP';
return $var;
?>
```

Ilustração : ex_return_inc.php

```php
<?php
$var = 'PHP';
?>
```

Ilustração : ex_return_inc_2.php

```php
<?php
$teste = include ' ex_return_inc.php';
echo $teste; // imprime 'PHP'
$teste2 = include ' ex_return_inc_2.php';
echo $teste2; // imprime 1
?>
```

Ilustração : ex_return_inc_3.php

$teste2 assimila o valor **1** porque a inclusão foi realizada com sucesso. Verifique a diferença entre os exemplos. O primeiro utiliza **return( )** dentro do arquivo incluído enquanto que o outro não. Há outras maneiras de "incluir" arquivos dentro de variáveis, com fopen( ), file( ) ou utilizando include( ).

Funções de Data

Uma das tarefas mais comum, em se tratando de PHP, é o uso de funções para trabalhar com Datas. Aqui veremos as mais comuns.

Date

A função date() é utilizada para formatar data e hora locais (do servidor onde o PHP está rodando).

Tabela com os caracteres utilizados junto com a função.

Caractere de formato	*Descrição*	*Exemplo de valores retornados*
`A`	Retorna Ante meridiem e Post meridiem em minúsculo	`am` or `pm`
`A`	Retorna Ante meridiem e Post meridiem em maíusculo	`AM` ou `PM`
`d`	Dia do mês, 2 digitos com leading zeros	`01` to `31`
`D`	Uma representação textual de um dia, três letras	`Mon` through `Sun`
`F`	Uma representação textual de um mês, tal como janeiro ou Março	`Janeiro` a `Dezembro`
`g`	12-hour formato de hora sem zeros	`1` a `12`
`G`	24-hour formato de hora sem zeros	`0` a `23`
`h`	12-hour formato de hora com zeros	`01` a `12`
`H`	24-hour formato de hora com zeros	`00` a `23`
`i`	Minutos com zeros	`00` para `59`
`I` (i maiúsculo)	Se a data está ou não em horário de verão	`1` Horário de Verão, `0` sem Horário de Verão.
`j`	Dia do mês sem leading zeros	`1` to `31`
`l` ('L' minúsculo)	Uma representação completa textual de um dia da semana	`Domingo` a `Sábado`
`L`	Se é ano bissexto	`1` ano bissexto, `0` ano não bissexto.

m	Representação numérica de um mês, com leading zeros	01 a 12
M	Uma representação textual curta de um mês, três letras	Jan a Dec
n	Representação numérica de um mês, sem leading zeros	1 a 12
O	Diferença ao horário de Greenwich (GMT) em horas	Exemplo: +0200
r	*RFC 2822* formatted date	Exemplo: Thu, 21 Dec 2000 16:01:07 +0200
s	Segundos, com leading zeros	00 a 59
S	Sufixo ordinal inglês para o dia do mês, 2 caracteres	st, nd, rd ou th. Funciona bem com j
t	Número de dias do dado mês	28 a 31
T	Timezone setting of this machine	Examples: EST, MDT ...
U	Seconds since the Unix Epoch (January 1 1970 00:00:00 GMT)	See also **time()**
w	Representação numérica do dia da semana	0 (para Domingo) a 6 (para Sábado)
W	Número da semana do ano ISO-8601, semanas começadas na segunda-feira (adicionado no PHP 4.1.0)	Exemplo: 42 (a 42nd (quadragésima segunda) semana do ano)
Y	Uma representação completa do ano, 4 dígitos	Examples: 1999 ou 2003
y	Representação do ano em dois dígitos	Examples: 99 ou 03
z	O dia do ano (começando de 0)	0 a 365
Z	Timezone offset in seconds. The offset for timezones west of UTC is always negative, and for those east of UTC is always positive.	-43200 até 43200

Vejamos sua utilização:

```php
<?php
$dataLocal = date("d/m/Y H:i:s");
echo $dataLocal; // mostra data e hora local
?>
```

Getdate

A função getdate() retorna a data e hora local, em formato de Array. Abaixo, a tabela com as respectivas chaves retornadas no array:

Chave	*Descrição*	*Exemplo dos valores retornados*
"seconds"	Representação numérica dos segundos	0 a 59
"minutes"	Representação numérica dos minutos	0 a 59
"hours"	Representação numérica das horas	0 a 23
"mday"	Representação numérica do dia do mês	1 a 31
"wday"	Representação numérica do dia da semana	0 (para Sunday) a 6 (para Saturday)
"mon"	Representação numérica de um mês	1 a 12
"year"	Representação numérica completa do ano, 4 digitos	Exemples: 1999 ou 2003
"yday"	Numeric representation of the day of the year	0 a 366
"weekday"	Representação textual completa do dia da semana	Sunday a Saturday

"month"	Representação textual completa de um mês, tal como January ou March	January a December
0	Segundos desde a época UNIX, similar aos valores retornados por **time()** e usados por **date()**.	System Dependent, tipicamente de -2147483648 a 2147483647.

```php
<?php
$hoje = getdate();
echo $hoje['year']; // imprime o ano atual
?>
```

Vejamos sua utilização:

Time

A função time() retorna o timestamp Unix atual.Um timestamp é a hora atual medida no número de segundos desde a Era Unix (01/01/1970).

```php
<?php
$tsAtual = time();
echo $tsAtual; // imprime timestamp atual na tela
?>
```

Exemplo:

Mktime

A função mktime() é utilizada para se retornar um timestamp Unix (formato utilizado para datas no PHP) a partir de uma data qualquer maior que 01/01/1970.

```php
<?php
mktime (hora, minuto, segundos, mes, dia, ano)
?>
```

Sua sintaxe é simples:

Depois de obtido o timestamp, você pode utilizá-lo com as funções previamente citadas, a exemplo da função date.

Veja um exemplo:

```php
<?php
// obtem timestamp de 01/04/2006 às 00:00:00
$timeStamp = mktime(0, 0, 0, 4, 1, 2006);
// retorna o dia da semana em inglês, usando a função date
$data = date("l", $timeStamp);

// imprime o dia da semana referente a data 01/04/2006
echo $data;
?>
```

Formulários

Os formulários são recursos oferecidos desde o HTML que, dentre outras funcionalidades, tem o intuito de interagir com o usuário. Seja em forma de enquetes, contato, pesquisas, etc.

Métodos GET e POST

Para se trabalhar com formulários existem 2 tipos de métodos utilizados para o tráfego dos dados: GET e POST.

O primeiro deles, o método GET, trafega os dados de forma visível ao usuário, na barra de endereços do navegador (browser). É útil para fazer marcação de URLs, como exemplo de sites de busca, onde o usuário deseja salvar a página para consulta posterior.

O método POST, trafega os dados de forma oculta, onde somente o navegador (browser) e o servidor de onde foi postado os dados, tem acesso à leitura destes. É utilizado para tráfego de dados de pesquisas por exemplo, onde o usuário não usará novamente a página para favoritar, etc.

Esses métodos são definidos na tag <form> do HTML:

```html
<form method="post" action="arquivo.php"></form>
```

No lado do PHP, os dados são tratados utilizando-se de variáveis globais. Para o método GET é usado a variável $_GET, e no método POST, a variável $_POST.

Existe ainda a variável $_REQUEST, que associa automaticamente o método recebido (GET ou POST). Todas estas variáveis são tratadas como arrays, onde a chave é o nome do campo do formulário. Assim se você tem um campo chamado nome, no PHP pode ser apontado como $_REQUEST[‘nome’] .

A diferença entre GET e POST está basicamente na capacidade do envio de dados, uma vez que GET é limitado à cerca de 2KB de dados e POST não possui limite. Outra diferença está na forma de envio. GET envia os dados anexados ao nome do script informado (utiliza-se para isso o símbolo ? após o nome do script, seguido da relação de campos e seus respectivos valores separados por &). Já no método POST os dados são enviados no corpo da mensagem que será enviada no servidor.

Formulários na prática

Nesta seção veremos na prática um exemplo de um formulário completo, com uso dos tipos texto, senha, radio, etc.

Uma observação importante: se você tem vários campos com o mesmo nome e de múltipla escolha no formulário (como é o caso dos checkbox’s), esses campos serão tratados como array do array global. O exemplo a seguir é figurativo (uma pesquisa de gostos por livros), e será usado como base para você

aprender a manusear dados de um formulário.

Exemplo de um formulário HTML:

```
<form action="envia.php" method="post">
<fieldset>
<legend>Dados Pessoais</legend>
<label>Nome:</label><br />
<input type="text" name="nome" size="30" /><br />
<label>Endereço:</label><br />
<input type="text" name="endereco" size="30" /><br />
</fieldset>
<fieldset>
<legend>Pesquisa</legend>
<label>Quantos livros você compra por ano?</label><br />
<input type="radio" name="livros" value="0" />Nenhum<br />
<input type="radio" name="livros" value="1-5" />De 1 a 5<br />
<input type="radio" name="livros" value="5-10" />De 5 a 10<br />
<input type="radio" name="livros" value="10-20" />De 10 a 20<br />
<input type="radio" name="livros" value="20+" />Mais de 20<br />
<label>Qual seu estilo de livro favorito?</label><br />
<input type="checkbox" name="estilo[]" value="Romance" />Romance<br />
<input type="checkbox" name="estilo[]" value="Drama" />Drama<br />
<input type="checkbox" name="estilo[]" value="Suspense" />Suspense<br />
<input type="checkbox" name="estilo[]" value="Tecnico" />Técnico<br />
</fieldset>
<input type="submit" />
</form>
```

Exemplo do arquivo envia.php, que será submetido pelo formulário:

```php
<?php
echo "Nome: ".$_POST['nome'] . "<br />";
echo "Endereço: ".$_POST['endereco'] . "<br />";
echo "Qt. Livros: ".$_POST['livros']. "<br />";
echo "Estilo: ";
foreach($_POST['estilo'] as $estilo)
{
   echo $estilo. " - ";
}
?>
```

Perceba que já fizemos uso do construtor foreach acima, pois o campo estilo[ ] do formulário é múltipla-escolha, logo, ele é considerado um array no PHP. Repare ainda que o campo possui os colchetes []. Isso precisa ser determinado para que o PHP entenda como um Array.

Acessando o MySQL via PHP

Para acessar bases de dados num servidor MySQL, é necessário antes estabelecer uma conexão. Para isso, deve ser utilizado o comando mysql_connect, ou o mysql_pconnect. A diferença entre os dois comandos é que o mysql_pconnect estabelece uma conexão permanente, ou seja, que não é encerrada ao final da execução do script. As assinaturas dos dois comandos são semelhantes, como pode ser verificado a seguir:

```
int mysql_connect(string [host[:porta]] , string [login] , string [senha] );
int mysql_pconnect(string [host[:porta]] , string [login] , string [senha] );
```

O valor de retorno é um inteiro que identifica a conexão, ou falso se a conexão falhar. Antes de tentar estabelecer uma conexão, o interpretador PHP verifica se já existe uma conexão estabelecida com o mesmo host, o mesmo login e a mesma senha. Se existir, o identificador desta conexão é retornado. Senão, uma nova conexão é criada.

Uma conexão estabelecida com o comando mysql_connect é encerrada ao final da execução do script. Para encerrá-la antes disso deve ser utilizado o comando mysql_close, que tem a seguinte assinatura:

```
int mysql_close(int [identificador da conexão] );
```

Se o identificador não for fornecido, a última conexão estabelecida será encerrada.

IMPORTANTE: o comando mysql_close não encerra conexões estabelecidas com o comando mysql_pconnect.

Selecionando a base de dados

Depois de estabelecida a conexão, é preciso selecionar a base de dados a ser utilizada, através do

comando mysql_select_db, que segue o seguinte modelo:

```
int mysql_select_db(string base, int [conexao] );
```

Novamente, se o identificador da conexão não for fornecido, a última conexão estabelecida será utilizada.

Realizando consultas

Para executar consultas SQL no MySQL, utiliza-se o comando mysql_query, que tem a seguinte assinatura:

```
int mysql_query(string query, int [conexao] );
```

Onde query é a expressão SQL a ser executada, sem o ponto-e-vírgula no final, e conexao é o identificador da conexão a ser utilizada. A consulta será executada na base de dados selecionada pelo comando mysql_select_db.

É bom lembrar que uma consulta não significa apenas um comando SELECT. A consulta pode conter qualquer comando SQL aceito pelo banco.

O valor de retorno é falso se a expressão SQL for incorreta, e diferente de zero se for correta. No caso de uma expressão SELECT, as linhas retornadas são armazenadas numa memória de resultados, e o valor de retorno é o identificador do resultado. Alguns comandos podem ser realizados com esse resultado:

Apagando o resultado

```
int mysql_free_result(int result);
```

O comando mysql_free-result deve ser utilizado para apagar da memória o resultado indicado.

Número de linhas

```
int mysql_num_rows(int result);
```

O comando mysql_num_rows retorna o número de linhas contidas num resultado.

Utilizando os resultados

Existem diversas maneiras de ler os resultados de uma query SELECT. As mais comuns serão vistas a seguir:

```
int mysql_result(int result, int linha, mixed [campo] );
```

Retorna o conteúdo de uma célula da tabela de resultados.

- result é o identificador do resultado;
- linha é o número da linha, iniciado por 0;
- campo é uma string com o nome do campo, ou um número correspondente ao número da coluna. Se foi utilizado um alias na consulta, este deve ser utilizado no comando mysql_result.

Este comando deve ser utilizado apenas para resultados pequenos. Quando o volume de dados for maior, é recomendado utilizar um dos métodos a seguir:

```
array mysql_fetch_array(int result);
```

Lê uma linha do resultado e devolve um array, cujos índices são os nomes dos campos. A execução seguinte do mesmo comando lerá a próxima linha, até chegar ao final do resultado.

```
array mysql_fetch_row(int result);
```

Semelhante ao comando anterior, com a diferença que os índices do array são numéricos, iniciando pelo 0 (zero).

Alterando o ponteiro de um resultado

```
int mysql_data_seek(int result, int numero);
```

Cada resultado possui um “ponteiro”, que indica qual será a próxima linha lida com o comando mysql_fetch_row (ou mysql_fetch_array). Para alterar a posição indicada por esse ponteiro deve ser utilizada a função mysql_data_seek, sendo que o número da primeira linha de um resultado é zero.

Sessões e Cookies

Neste capítulo vamos estudas duas das funcionalidades mais utilizadas no PHP. Ambas são utilizadas para armazenar dados por um período e diferem no sentido de que as Sessões expiram quando o usuário fecha o browser, e os Cookies podem perdurar um tempo definido pelo programador.

Uma das utilizações destas funcionalidades são por exemplo, os carrinho de compras, que armazenam as compras feitas pelo usuário. Outro exemplo é o de criar função de login, área restrita de acesso, etc.

Vamos estudar cada uma separadamente.

Sessões

As sessões são forma simples de armazenar dados temporários. Os dados são apagados assim que o usuário fechar o browser. Mas, é importante que você saiba, que você pode excluir sessões quando quiser.

Imagine o seguinte problema: você tem um site, e quer que em algumas seções, apenas usuários que saibam uma determinada senha consigam acessar. Você pode utilizar sessões neste caso. Vamos ver como?

O ponto de partida para o uso de sessões, é com sua inicialização, fazendo uso da função session_start(). Vale lembrar, que até usá-la, não se pode imprimir nada na tela (seja via echo, print, avisos ou erros de algum trecho de código).

Por estes motivos, recomenda-se que você insira a função logo na primeira linha do seu script. Sempre que for utilizar sessões, você vai precisar inicializar esta função.

Exemplo:

```php
<?php
session_start();
?>
```

Dentro da seção, fazemos uso da variável global $_SESSION para definir variáveis e valores.Agora voltemos ao problema inicial. Precisamos restringir acesso a algumas seções. Vamos fazer isso, utilizando uma variável de sessão para saber se o usuário passou pela tela de login e informou a senha corretamente.

Nas seções onde a área será restrita, fazemos a verificação se o usuário logou.

```php
<?php
session_start();
if (!$_SESSION['logado'])
{
   header("Location: login.htm");
   die;
}

?>
```

Veja o exemplo da pagina_secreta.php:

Entendendo o código: primeiro inicializamos a sessão com a função session_start(). Em seguida fazemos a verificação: se não existir (uso da !) a variável de sessão “logado” ou se ela for igual a falso (booleano), redirecionamos para a página de login (função header) e finalizamos o script (função die). Caso contrário, o script prossegue, e a sessão é mostrada normalmente.

Vamos entender agora a página de login. Primeiro vamos criar o formulário do login.htm:

```html
<form action="login.php" method="post">

<fieldset>
<legend>Informe a senha</legend>
   <label>Senha:</label>
   <input type="password" name="senha" size="10" />
</fieldset>

</form>
```

Agora vejamos o arquivo login.php:

```php
<?php
session_start();
 if ($_POST['senha'] == 'secreta')
{
   $_SESSION['logado'] = true;
   header("Location: pagina_secreta.php");
   die;
}
else
{
   echo "Senha incorreta";
}
?>
```

Entendendo o código: primeiro é feita a inicialização da sessão (função session_start). Em seguida é feita a comparação, se a senha digitada é igual a “secreta” (pode ser alterada), em caso verdadeiro é definida a variável de sessão “logado” como verdadeiro e redireciona o usuário para a página secreta. Caso contrário, imprime na tela “senha incorreta”.

Cookies

Cookies, na livre tradução, são biscoitos.

Cookies são mecanismos para armazenar e consultar informações nos browsers dos visitantes da página. O PHP atribui cookies utilizando a função setcookie, que deve ser utilizada antes de qualquer impressão na página (mesmo esquema das sessões).

O uso de cookies não é recomendado quando se trata de informações sigilosas. Os dados dos cookies são armazenados no diretório de arquivos temporários do visitante, sendo facilmente visualizado por pessoas mal intencionadas.

A diferença entre sessões e cookies, é que os cookies podem ser acessados mesmo depois de o usuário ter fechado seu browser.

Outra questão importante, é que a opção “aceitar cookies” (do navegador – browser) que pode ser desativada a qualquer momento pelo visitante. Para uma transmissão de dados segura é recomendável o uso de sessões.

Sintaxe básica:

```
setcookie(“nome_do_cookie”,”seu_valor”,”tempo_de_vida”,”path”,”domínio”,”conexão_segura”)
```

- Nome_do_cookie = É o nome que, posteriormente, se tornará a variável e o que o servirá de referência para indicar o cookie.
- Seu_valor = É o valor que a variável possuirá. Esse valor pode ser de todos os tipos.
- Tempo_de_vida = É o tempo, em segundos, que o cookie existirá no computador do visitante. Uma vez excedido esse prazo o cookie se apaga de modo irrecuperável. Se esse argumento ficar vazio, o cookie se apagará quando o visitante fechar o browser.

- Path = endereço da página que gerou o cookie – automático
- Domínio = domínio ao qual pertence o cookie – automático
- Conexão_segura = Indica se o cookie deverá ser transmitido somente em uma conexão segura HTTPS.

Depois de definida um cookie, você pode acessá-lo através da variável global $_COOKIE[‘nome_do_cookie’].

Fontes Bibliográficas

http://www.icmc.usp.br/ensino/material/html/

http://maujor.com/tutorial/insetut.php

Hino Nacional

Ouviram do Ipiranga as margens plácidas
De um povo heróico o brado retumbante,
E o sol da liberdade, em raios fúlgidos,
Brilhou no céu da pátria nesse instante.

Se o penhor dessa igualdade
Conseguimos conquistar com braço forte,
Em teu seio, ó liberdade,
Desafia o nosso peito a própria morte!

Ó Pátria amada,
Idolatrada,
Salve! Salve!

Brasil, um sonho intenso, um raio vívido
De amor e de esperança à terra desce,
Se em teu formoso céu, risonho e límpido,
A imagem do Cruzeiro resplandece.

Gigante pela própria natureza,
És belo, és forte, impávido colosso,
E o teu futuro espelha essa grandeza.

Terra adorada,
Entre outras mil,
És tu, Brasil,
Ó Pátria amada!
Dos filhos deste solo és mãe gentil,
Pátria amada,Brasil!

Deitado eternamente em berço esplêndido,
Ao som do mar e à luz do céu profundo,
Fulguras, ó Brasil, florão da América,
Iluminado ao sol do Novo Mundo!

Do que a terra, mais garrida,
Teus risonhos, lindos campos têm mais flores;
"Nossos bosques têm mais vida",
"Nossa vida" no teu seio "mais amores."

Ó Pátria amada,
Idolatrada,
Salve! Salve!

Brasil, de amor eterno seja símbolo
O lábaro que ostentas estrelado,
E diga o verde-louro dessa flâmula
- "Paz no futuro e glória no passado."

Mas, se ergues da justiça a clava forte,
Verás que um filho teu não foge à luta,
Nem teme, quem te adora, a própria morte.

Terra adorada,
Entre outras mil,
És tu, Brasil,
Ó Pátria amada!
Dos filhos deste solo és mãe gentil,
Pátria amada, Brasil!

Hino do Estado do Ceará

Poesia de Thomaz Lopes
Música de Alberto Nepomuceno
Terra do sol, do amor, terra da luz!
Soa o clarim que tua glória conta!
Terra, o teu nome a fama aos céus remonta
Em clarão que seduz!
Nome que brilha esplêndido luzeiro
Nos fulvos braços de ouro do cruzeiro!

Mudem-se em flor as pedras dos caminhos!
Chuvas de prata rolem das estrelas...
E despertando, deslumbrada, ao vê-las
Ressoa a voz dos ninhos...
Há de florar nas rosas e nos cravos
Rubros o sangue ardente dos escravos.
Seja teu verbo a voz do coração,
Verbo de paz e amor do Sul ao Norte!
Ruja teu peito em luta contra a morte,
Acordando a amplidão.
Peito que deu alívio a quem sofria
E foi o sol iluminando o dia!

Tua jangada afoita enfune o pano!
Vento feliz conduza a vela ousada!
Que importa que no seu barco seja um nada
Na vastidão do oceano,
Se à proa vão heróis e marinheiros
E vão no peito corações guerreiros?

Se, nós te amamos, em aventuras e mágoas!
Porque esse chão que embebe a água dos rios
Há de florar em meses, nos estios
E bosques, pelas águas!
Selvas e rios, serras e florestas
Brotem no solo em rumorosas festas!
Abra-se ao vento o teu pendão natal
Sobre as revoltas águas dos teus mares!
E desfraldado diga aos céus e aos mares
A vitória imortal!
Que foi de sangue, em guerras leais e francas,
E foi na paz da cor das hóstias brancas!

GOVERNO DO
ESTADO DO CEARÁ

Secretaria da Educação