



Folhas de Estilo

4.1. Introdução

4.1.1. O que são folhas de estilo?

Uma folha de estilos é um conjunto de regras que informa a um programa, responsável pela formatação de um documento, como organizar a página, como posicionar e expor o texto e, dependendo de onde é aplicada, como organizar uma coleção de documentos.

A maior parte dos programas de editoração eletrônica e processadores de texto modernos trabalham com folhas de estilos. O processo consiste em definir um rótulo (nome do estilo) para um determinado parágrafo e em seguida alterar os seus atributos. Todo parágrafo que for rotulado com aquele estilo passará a exibir as características definidas anteriormente. Qualquer alteração nos atributos de um estilo afetará todos os parágrafos que estiverem rotulados com ele.

Esta descrição, que se aplica a estilos em processadores de texto e programas de editoração eletrônica, também vale para a Web. Na Web, os "parágrafos" são blocos marcados por descritores HTML como <H1>, <P>, etc. Para fazer com que todos os blocos de textos marcados com <H1> em um documento sejam exibidos em tamanho de 48 pontos, basta definir a regra:

```
H1 {font-size: 48pt}
```

dentro de uma "folha de estilos" aplicada ao documento.

A folha de estilos pode ser um arquivo de textos simples (alfabeto ISO-Latin1) com a extensão .css. Para vinculá-lo a uma página HTML, esta deve ter dentro do seu bloco <HEAD> ... </HEAD> o seguinte descritor:

```
<LINK REL="stylesheet" HREF="url_do_arquivo.css">
```

O restante deste artigo tratará dos fundamentos da tecnologia de folhas de estilos aplicáveis ao HTML, chamada de *Cascading Style Sheets* (folhas de estilo em cascata), mostrando como estabelecer as regras de estilo para um bloco de texto, uma página ou todo um site. Seções específicas abordarão cores, imagens, tipologia e posicionamento. Este texto não é completo. Omitimos propriedades e recursos não suportados nos browsers e nos limitamos àqueles recursos que constam da especificação CSS1 (não incluímos recursos proprietários nem a maior parte das novidades do CSS2 que não funcionam nos browsers disponíveis no mercado). Para uma abordagem mais completa, consulte a documentação oficial do W3C: <http://www.w3.org/Style/>.

4.1.2. Para que servem as folhas de estilo

Separar apresentação da estrutura

Com isto é possível voltar a suportar browsers antigos que antes estavam condenados por não conseguirem ler a informação sem perdas. Com a informação toda armazenada no HTML (estrutura), a apresentação (estilo) seria uma camada a mais, alterando a disposição do texto, cores, etc. mas sem afetar a estrutura essencial da informação. Isto permite que uma página tenha vários estilos e use *scripts* (programas embutidos) para decidir qual carregar (em função do browser e da plataforma). Isto é muito menos trabalho que fazer uma página para cada browser e plataforma, pois a atualização é feita apenas no HTML. Também, com isso, é possível ter uma folha de estilos especial somente para impressão, onde haveria informações de quebra de páginas, etc. (este recurso não faz parte da versão 1 do CSS).

Controle absoluto da aparência da página.

É algo que não se tem com o HTML. Pode-se usar tabelas, GIFs invisíveis de um pixel e mais uma dúzia de "macetes" mas não se consegue fazer o texto fluir suavemente em volta de uma imagem irregular, por exemplo. Além do mais, quanto mais sofisticada a técnica, mais difícil é de codificar e mais "sujo" fica o código, o que o torna mais sujeito a erros. Com CSS, pode-se colocar uma imagem em qualquer lugar da página (até fora dela), usando técnicas de posicionamento absoluto ou relativo. Pode-se escolher a posição exata da imagem de *background* e fazê-la combinar com algo no *foreground*. As dimensões e posições são exatas e dadas em unidades conhecidas no mundo da tipografia como pixels, pontos, *paucas*, milímetros.

Páginas mais leves

Com folhas de estilo é possível criar muitas páginas com um layout sofisticado que antes só era possível usando imagens, tecnologias como *Flash* ou applets Java. Estas páginas eram sempre mais pesadas, pois precisavam carregar imagens, componentes, programas. Com CSS é possível definir texto de qualquer tamanho, posicioná-lo por cima de outros objetos, ao lado ou por cima de texto e conseguir efeitos sofisticados a um custo (banda de rede) baixo. É possível ainda importar fontes (que o usuário talvez não tenha).

Manutenção de um grande site

Uma folha de estilos serve para toda uma coleção de páginas, podendo ser usada para dar um estilo consistente a todo o site. Sendo aplicada em separado da informação e estrutura, não precisará ser atualizada todas as vezes em que a informação for mudada. A página pode ser atualizada em um editor HTML ou gerador de HTML simples, sem recursos de cor ou alinhamento, e ser formatada na hora em que for carregada pelo browser. É possível também fazer o contrário: mudar o estilo sem alterar a informação, como ter uma página que sempre carrega com um estilo diferente.

O uso das folhas de estilo depende da boa estrutura do HTML. A linguagem CSS (é uma linguagem declarativa) trabalha com os elementos tratando-os como "objetos". Cada parágrafo <P>, cada <H1>, cada é um objeto. Objetos podem ser agrupados de várias formas. A cada objeto ou grupo de objetos podem ser atribuídas propriedades de estilo definidas em regras. Por exemplo, considere a seguinte regra: "todo objeto P da classe 'editorial' será azul, terá tamanho de 12 pontos, espaçamento duplo, alinhamento pela direita e usará a família de fontes Arial, ou, se esta não existir, Helvetica, ou então a fonte sem-serifa *default* do sistema". Um arquivo CSS com apenas a regra acima conteria o texto:

```
P.editorial {color: 0000ff;
             font-size: 12pt;
             line-height: 24pt;
             text-align: right;
             font-family: arial, helvetica, sans-serif}
```

Se a folha de estilos acima for aplicada a uma página que possua parágrafos <P> rotulados com o nome "editorial" (atributo 'CLASS=editorial'), eles serão formatados de acordo com as propriedades especificadas se o browser suportar CSS. Se o browser não suportar CSS, a estrutura será mantida (embora a aparência não seja a ideal) e o usuário conseguirá ter acesso à informação estruturada, mesmo em um meio de visualização mais limitado.

Quando se usa CSS, são poucas as modificações necessárias no HTML. Não são necessários novos descritores ou extensões. No exemplo acima, teremos que incluir apenas um atributo a mais (o atributo CLASS, do HTML 4) classificando os parágrafos que fazem parte do nosso 'editorial' (parágrafos que tem uma função diferente dos demais).

A grande vantagem das folhas de estilo é a preservação da estrutura do HTML e um controle muito melhor do autor sobre a sua aparência na tela do usuário final. Uma página deverá aparecer da melhor forma possível tanto num *PowerPC* sofisticado como naquele IBM PCXT 4.77MHz rodando o *Lynx* for DOS. O primeiro utilizará todos os recursos gráficos determinados pelas folhas de estilo. O segundo as ignorará, mas preservará a estrutura e a informação

4.2. Regras Básicas

4.2.1. Regras, declarações e seletores

A estrutura dos estilos é bastante simples. Consiste de uma *lista de regras*. Cada regra possui um bloco, entre chaves ({ e }), de uma ou mais declarações aplicáveis a um ou mais *seletores*. Um seletor é algo no qual pode-se aplicar um estilo. Pode ser um descritor HTML, uma hierarquia de descritores ou um atributo que identifique um grupo de descritores. Uma *folha de estilos* consiste de uma ou mais linhas de regras, da forma:

```
seletores { declarações }
```

As regras podem estar dentro de um arquivo de texto (ISO Latin1 ou ASCII 8-bit) com extensão ".css" ou *embutidas* em um arquivo HTML (as várias maneiras de aplicar estilos a um arquivo HTML serão vistas na seção seguinte).

Um exemplo de folha de estilos com apenas uma regra foi mostrada na seção anterior:

```
H1 {font-size: 48pt}
```

Nesta regra, H1 é o seletor e {font-size: 48pt} é o bloco da declaração, que estabelece um tamanho de fonte (prop. font-size) para todos os objetos (parágrafos) marcados com <H1>.

As declarações são feitas usando a sintaxe:

```
propriedade: valor
```

Observe que se usa dois-pontos (:) e não igual (=) para aplicar um valor a uma propriedade. Pode haver mais de uma declaração de estilo para um seletor. Isto pode ser feito em outro bloco. Cada linha acrescenta ou sobrepõe declarações feitas em linhas anteriores:

```
H1 { font-size: 24pt }
H1 { color: blue }
H1 { font-size: 18pt }
```

No trecho acima, o texto marcado com <H1> será azul e terá tamanho de 18pt porque a regra H1 { font-size: 18pt } ocorreu depois da regra H1 { font-size: 24pt }.

4.2.2. Múltiplas declarações e seletores

Várias declarações de estilo podem ser aplicadas de uma vez a um seletor. As declarações, então, precisam ser separadas por ponto-e-vírgula (;) :

```
H1 {font-size: 18pt; color: blue; font-family: Caslon, serif }
BODY { background : navy; color : white }
```

Os espaços em branco (espaços, novas-linhas e tabulações) são ignorados pelo browser na hora de interpretar uma folha de estilos e podem ser usados para melhorar a sua legibilidade. As duas linhas acima ficariam mais legíveis da forma:

```
BODY {background : navy;
      color : white }

H1   {color: blue;
      font-size: 18pt;
      font-family: Caslon, serif }
```

Uma declaração só termina com uma fecha-chaves (}) ou com um ponto-e-vírgula (;). Dependendo da propriedade, esta pode ter vários valores separados por vírgulas ou valores compostos com as palavras separadas por espaços:

```
P { font: 12pt "Times New Roman" bold }
H2 { font-family: Arial, Helvetica, Sans-serif }
```

As aspas devem ser usadas quando uma palavra que tem espaços precisar ser usada. No exemplo acima, o nome "Times New Roman" deveria ser tratado como o nome de uma fonte distinta, e não como três valores, o que ocorreria se as aspas não estivessem presentes.

Assim como um seletor pode ter várias propriedades definidas para ele, um mesmo conjunto de propriedades pode ser aplicada a um grupo de seletores, separando-os com vírgulas:

```
H1, H2, H3 { color: blue;
             font-size: 18pt;
             font-family: Arial, Helvetica, Sans-serif }
```

As declarações de estilo podem ser aplicadas em quase qualquer descritor HTML - no mundo perfeito! Na prática, muitos browsers ainda têm problemas de compatibilidade, e não implementam a especificação à risca, como veremos adiante.

Ao utilizar folhas de estilo, deve-se respeitar os elementos HTML que possuem descritores finais freqüentemente ignorados, como `</P>`, ``, etc. A falta do `</P>` pode causar o "vazamento" das declarações de estilo para fora do parágrafo em alguns browsers.

4.2.3. Comentários e instruções

Além das regras, um arquivo CSS pode ter ainda comentários e instruções (precedidas de `@`). No CSS1 apenas uma instrução é definida: `@import`. Ela é usada para que uma folha de estilos possa importar estilos de outro arquivo CSS através de uma URL. Os estilos importados sempre têm menos precedência que os locais (ou seja, os locais podem sobrepor os importados). A sintaxe da instrução `@import` é:

```
@import url(url_da_folha_de_estilos)
```

Não deve haver outras estruturas (a não ser comentários) na linha onde há uma instrução. Exemplos do uso de `@import`:

```
@import url(../basico.css)
@import url(http://longe.com/estilos/basico.css)
```

Pode-se inserir trechos que serão ignorados pelo browser ao interpretar folhas de estilo usando blocos de comentário. Comentários em folhas de estilos são iguais a comentários em linguagens como C ou Java: entre `/*` e `*/`:

```
/* este texto é ignorado até que seja encontrado
   um asterisco seguido por uma barra */
```

4.2.4. Valores

Os valores que são aplicados às propriedades têm uma sintaxe que depende da propriedade que os usa. Propriedades que envolvem tamanho (tamanho de fontes, espaçamento, etc.) geralmente recebem valores que consistem de um número e uma unidade ou porcentagem. O

sinal de porcentagem ou unidade deve estar junto ao número correspondente sem espaços. Ou seja, deve-se escrever `font-size: 24pt` e não `font-size: 24 pt`.

Cores e arquivos externos podem requerer uma função para serem definidos. São duas as funções (ou procedimentos) do CSS1: `rgb()`, que constrói uma cor, e `url()`, que retorna um vínculo para uma imagem ou arquivo CSS (usada em instruções `@import`).

Há quatro maneiras diferentes de especificar cores em CSS: usando o nome do sistema (red, yellow, blue, black, lightGray), usando seu código RGB hexadecimal (ff0000, ffff00, 0000ff, 34adfc, 80a7a7) ou usando a função `rgb()`. A função `rgb()` requer três argumentos que representam a intensidade dos componentes vermelho (R), verde (G) e azul (B) de uma cor em forma de luz (não opaca). A intensidade pode ser expressa em valores inteiros de 0 (mínimo) a 255 (máximo) ou em valores fracionários de 0% a 100%. As instruções abaixo definem a mesma cor para um parágrafo:

```
P {color: red}
P {color: ff0000}
P {color: rgb(100%, 0%, 0%)}
P {color: rgb(255, 0, 0)}
```

Não deve haver espaço entre o "b" de `rgb` e o abre-parênteses.

A função URL pode ser usada em propriedades que requerem arquivos (no caso, imagens) como valores. Ela recebe um argumento apenas com a URL (relativa ou absoluta) da imagem:

```
P {background-image: url(../imagens/tijolos.gif)}
P {background-image: url(http://longe.com/imagens/pedras.png)}
```

4.2.5. Herança

Os estilos "herdam" propriedades de várias maneiras. Uma das formas é através da própria hierarquia do HTML. Se você declara propriedades para BODY, todos os descritores serão afetados a não ser que tenham as suas propriedades redefinidas dentro de um novo bloco de declarações CSS. Se um <I> está dentro de um <P> e todos os <P> são declarados como tendo a cor vermelha, o <I> também será vermelho a menos que haja um bloco, posterior àquela declaração, redefinindo as propriedades de <I>, por exemplo:

```
P {font: 12pt "Times New Roman" bold;
   color: red }

I {color: black }
```

faria com que o texto "seletor", no texto a seguir permanecesse preto:

```
<P>Um <I>seletor</I> é algo no qual pode-se aplicar um estilo.</P>
```

Se você definir atributos para os descritores <BODY> ou <HTML>, toda a página será afetada. No exemplo a seguir, uma cor de texto definida para BODY será usada para colorir todo o texto do documento, a não ser que sejam sobrepostos por uma regra subsequente:

```
BODY {color: navy }
H1, H2 {color: yellow }
```

Os blocos acima farão com que todo o texto seja azul marinho, exceto aquele marcado com H1 ou H2, que será amarelo.

Os browsers comerciais têm problemas principalmente com a aplicação de estilos em BODY, portanto, freqüentemente é preciso mexer nas declarações de estilo, acrescentando propriedades redundantes para adaptá-los à realidade. No site do W3C (<http://www.w3.org>) há links para documentos que analisam essas diferenças entre browsers. O site <http://www.w3.org/Style/CSS/Test/> é uma plataforma de testes que pode ser usada para verificar se um browser suporta ou não determinada propriedade.

4.2.6. *Descritores HTML especiais*

Dois descritores HTML têm importância fundamental em CSS. Eles são descritores estruturais puros que não definem apresentação específica na folha de estilos nativa do browser. Com CSS é possível definir propriedades de apresentação para esses descritores. Eles são <DIV> e .

 é um descritor que deve ser usado dentro de blocos de texto apenas. É chamado de descritor em-linha (*inline*), já que não quebra a linha antes ou depois. Ele se assemelha a descritores como , <I>, <SMALL>, <A HREF> e <SUP> que servem para formatar texto dentro de parágrafos, células de tabela, etc.

<DIV> é um descritor que define um bloco ou seção da página. Pode ser usado para dividir a página em seções (e subseções no *Internet Explorer*) e permitir que sejam aplicados estilos específicos a essas seções. Descritores de bloco são <P>, <H1>, <TABLE>, etc. <DIV> define um bloco sem função ou aparência definida. A função e aparência será determinada em CSS.

4.2.7. *Como incluir estilos em uma página*

Há três formas de aplicar uma folha de estilos a uma página Web. Estas formas irão determinar a abrangência dos estilos: se afetarão um trecho limitado de uma página, se a toda a página, ou se irão afetar todo um site.

A primeira forma, mais abrangente, é a vinculação a um arquivo CSS. Isto é feito ligando a página HTML a um arquivo de folha de estilo, usando do descritor <LINK>. Este método permite que múltiplas páginas ou um site inteiro usem a mesma folha de estilos.

Para fazer um vínculo à uma folha de estilos externa, deve-se criar um arquivo de texto contendo um conjunto de regras de estilo em CSS, salvá-lo com uma extensão ".css" e chamá-

lo a partir de todos os documentos HTML onde o estilo será aplicado. Não é preciso compilar ou qualquer coisa do tipo. Depois que as definições estiverem prontas, o estilo estará pronto para ser usado. Pode ser importado através do descritor LINK:

```
<HEAD>
( ... )
<LINK REL=STYLESHEET
      HREF="http://internet-name/mystyles.css"
      TYPE="text/css">
</HEAD>
```

O elemento <LINK> não tem descritor de fechamento e deve ser usado dentro do bloco <HEAD>.

Uma segunda forma, permite que estilos sejam aplicados localmente, em uma única página, embutindo uma folha de estilos diretamente na página HTML dentro de um bloco formado pelos descritores <STYLE> e </STYLE>. Este método permite que você altere as propriedades de estilo de uma única página.

A linguagem que é embutida em um bloco <STYLE> é a mesma usada nos arquivos CSS. <STYLE> ... </STYLE> deve ser usado em <HEAD>. Um atributo *type* informa o tipo de arquivo utilizado:

```
<style type="text/css">
  P { font: 12pt "Times New Roman" bold;
      color: red }
  I { color: black }
</style>
```

As propriedades declaradas no bloco <STYLE> sobrepõem qualquer propriedades anteriores do elemento (inclusive as de uma folha de estilos importada, se houver). É comum colocar todo o código entre comentários HTML (<!--e -->) para proteger browsers antigos da exibição indesejada do código:

```
<style type="text/css">
<!--
  P { font: 12pt "Times New Roman" bold;
      color: red }
  I { color: black }
-->
</style>
```

Finalmente, há uma forma de aplicar estilos diretamente a um descritor individual. Para fazer isto deve-se usar o atributo STYLE em quase qualquer descritor. Este método não corresponde exatamente a uma "folha" de estilos, pois os estilos aplicados não são reaproveitáveis.

Permite alterar a aparência de um único descritor, de um conjunto deles ou de um bloco de informações da página. Por exemplo:

```
<P STYLE="color: green; font-size: 12pt">Este texto</P>
```

Esta propriedade é mais interessante quando aplicada em um descritor que é usado para agrupar vários outros, como <DIV>, que divide a página em seções ou , usado em situações bem específicas. Usar estilos desta forma é pouco diferente de usar atributos locais. Os benefícios de poder mudar a fonte, cores e outras características em todo o documento ficam mais difíceis.

Pode-se usar um, dois ou os três métodos ao mesmo tempo. As características definidas pelos mais específicos sobrepõem as definidas pelos mais genéricos. Por exemplo, suponha que o arquivo `estilos.css` contenha apenas as seguintes regras:

```
H1 { color: green;
      font-size: 24pt}
P { color: blue}
```

Suponha que ele seja carregado na página a seguir que possui um bloco <STYLE> com uma nova definição de P e H1.

```
<HEAD>
<link rel=STYLESHEET
      href="estilos.css"
      type="text/css">
```

```
<style type="text/css"> <!--
    P {font: 12pt "Times New Roman" bold;
        color: red }
    H1 {color: black }
--> </style>
</HEAD>
```

Mais adiante, existe um parágrafo e um título da forma:

```
<h1 style="color: navy">Auto da Compadecida</h1>
<p style="color: black">Ariano Suassuna (Recife, 1955)</p>
```

Qual estilo irá predominar? Pela regra de que o mais específico sobrepõe o mais geral, teremos uma página onde os `<h1>` têm tamanho 24pt (do estilo importado), cor preta (valor sobreposto pelo estilo da página) e os `<p>` são vermelhos (sobreposto pelo estilo da página). Nas duas linhas acima (e nelas apenas), o `<h1>` será azul marinho (sobrepondo o estilo da página) e o `<p>` será preto.

4.2.8. Classes e IDs

Às vezes um parágrafo tem uma aparência diferente dos outros parágrafos em uma certa parte do texto. Para mudar o estilo dele, pode-se incluir as declarações em um atributo `STYLE`. Mas, se tal procedimento torna difícil a localização e a gerência dos estilos, pode-se usar um recurso para marcá-lo de forma que seja considerado diferente. Isto pode ser feito atribuindo-lhe uma identificação única. Em HTML 3.2, pode-se usar o atributo `ID`:

```
<P ID=w779>Texto especial</P>
```

Para alterar as características deste parágrafo agora, pode-se usar o seu `ID` como seletor, da forma:

```
#w779 {color: cyan }
```

Se isto estiver em um arquivo CSS, todas as páginas que o usam e que tiverem um elemento com o `ID #w779` serão afetadas. Se houver mais de um com o mesmo `ID` apenas o primeiro será afetado.

Melhor que usar `ID` é agrupar características semelhantes em classes. Uma classe é uma variação de um determinado objeto. Por exemplo, um texto teatral pode ter três parágrafos com apresentação diferente, representando as falas de três personagens. Se quiséssemos que cada um tivesse uma cor diferente, poderíamos declarar cada um como sendo de uma classe distinta:

```
<p class=padre>Eu retiro o que disse, João</p>
<p class=grilo>Retirando ou não retirando, o fato é que o cachorro
    enterrou-se em latim</p>
<p class=bispo>Um cachorro? Enterrado em latim? </p>
```

```
<p class=padre>Enterrado latindo, Senhor Bispo, Au, au, au, não sa-  
be? </p>
```

Para dar a cada parágrafo de um mesmo personagem (mesma classe) os mesmos atributos, usa-se:

```
P.grilo { color: maroon }  
P.padre { color: black }  
P.bispo { color: navy }
```

Desta maneira, todos os textos que deverão ser lidos pelo personagem "Bispo" estarão em azul marinho.

Uma classe também pode conter descritores diferentes. Se todos os textos citados por um certo autor tivessem que estar em outra cor ou fonte, poderíamos criar uma classe sem citar o descritor:

```
.verde { color: green }
```

Todos os descritores que tiverem o atributo CLASS=verde serão afetados, por exemplo: <P class=verde>, <h3 class=verde>, <table class=verde>, etc.

4.2.9. *Links (pseudo-classes e pseudo-elementos)*

Para seletores especiais que mudam de estado, como o texto marcado com <A>, é possível atribuir propriedades diferentes para cada estado:

```
A:link {color: red}  
A:active {color: 660011}  
A:visited {color: black; text-decoration: none}  
A:hover {color: blue; text-decoration: underline}
```

muda as características dos links não-visitados, ativos e visitados. Assim como qualquer seletor, os links podem ser combinados com outros descritores:

```
P, A:link, H2 {color: red}
```

4.2.10. *Seletores de contexto*

Você também pode definir seletores que só serão aplicados se estiverem no contexto de um outro seletor, por exemplo:

```
P.verde EM {color: 000040}
```

indica que o EM só terá sua cor alterada se ocorrer dentro de um bloco P da classe "verde".

Os seletores de contexto podem ser bem longos:

```
.bispo P UL UL LI A.classX:link {font-style: italic }
```

fará com que apenas os links não visitados da classe "classX" que estejam dentro de itens de lista de segundo nível situados dentro de um parágrafo dentro de um bloco qualquer da classe "bispo" sejam mostradas em itálico.

4.2.11. *Cascata de folhas de estilo*

Existem seis diferentes folhas de estilo que podem ser definidas. Além das três formas que mostramos em seção anterior deste capítulo, há ainda, segundo a especificação, mais três folhas de estilos que podem afetar uma página: 1) uma folha de estilos que é importada por outra folha de estilos (isto é diferente daquela que é vinculada ao HTML, dentro de um <link>), 2) uma folha de estilos definida pelo usuário (ou leitor da página) e 3) a folha de estilos *default* do browser (que é usada quando outra folha não define os estilos).

Todas estas folhas de estilo diferentes podem provocar uma grande confusão se não houver uma regra clara de como devem ser consideradas. Ainda há um sétimo fator que é a formatação introduzida pelo HTML, como nos descritores e atributos align=center. Listando todas as folhas de estilos que podem afetar um texto, temos:

1. *Folha de estilos default do browser*: todos os browsers possuem regras comuns para formatar um texto. A especificação HTML não impõe uma formatação padrão. *Netscape Navigator* por exemplo usa um fundo cinza como padrão e links azuis sublinhados. Já o *Internet Explorer* prefere um fundo branco.
2. *Folha de estilos definida pelo leitor*: a especificação define a possibilidade do leitor estabelecer uma folha de estilos própria. Isto é parcialmente conseguido quando o browser permite que se escolha diferentes cores para fundo, texto e links.
3. *Folha de estilos vinculada ao HTML*: é a folha de estilos que é importada pelo arquivo HTML através do descritor de ligação <link>
4. *Folha de estilos importada*: uma folha de estilos externa (arquivo CSS) pode ser importada de dentro de outra folha de estilos (um outro arquivo CSS ou bloco <style> no HTML) usando um comando especial @import:


```
@import url (outroestilo.css)
```
5. *Folha de estilos embutida no HTML*: é a folha de estilos que aparece na página HTML entre os descritores <style> e </style>.
6. *Folha de estilos local*: é aquela que é aplicada localmente a um descritor usando o atributo style="lista de declarações".
7. *Estilo definido pelo HTML*: atributos e descritores podem provocar alterações na aparência do texto, por exemplo: , <big>, <body bgcolor>, <p align=center>, etc.

Um browser que siga a especificação CSS à risca obedece a seguinte ordem de precedência: 1. Local, 2. Embutida, 3. Vinculada, 4. Importada, 5. HTML, 6. Leitor, 7. Browser

Na prática, as coisas não são tão bonitas. No *Internet Explorer* para *Macintosh*, a ordem é respeitada. Na versão do *Internet Explorer 3.0* para *Windows*, os estilos vinculados ao HTML têm

mais importância que os embutidos (o mesmo ocorre com *Explorer 4* e *Navigator 4*). No *Netscape Navigator 4* e *Internet Explorer 4*, os estilos aplicados via HTML têm precedência máxima (no *Internet Explorer 3* a precedência funciona corretamente, mas não no 4). Com as diferenças existentes nos browsers de hoje, não vale a pena ainda se aprofundar neste assunto. A solução ainda é testar, testar, testar antes de colocar no ar.

4.3. Fontes

Fontes são estilos de apresentação consistentes aplicados a alfabetos. Uma fonte consiste de atributos que alteram a aparência de um símbolo, sem alterar o seu significado. Oferecem as informações necessárias para que uma letra ou símbolo possa ser representado graficamente.

Os atributos essenciais de uma fonte são:

- Seu tipo (ou família)
- Seu tamanho
- Seu estilo (regular, itálico, *outline*, etc.)
- Seu peso (normal, negrito, *light*, *black*)

Para representar qualquer texto, portanto, é preciso escolher uma fonte, ou seja: um tipo, um estilo, um peso e um tamanho. Letras maiúsculas e minúsculas não são consideradas fontes diferentes, pois têm um significado distinto.

Os quatro atributos acima podem ser definidos em CSS através das propriedades `font-family`, `font-size`, `font-style` e `font-weight`. Não é preciso definir todas pois sempre têm valores *default*. CSS oferece ainda `font-variant`, que permite considerar outras variações de uma fonte.

4.3.1. *font-family*

Uma família de fontes (tipo) é selecionada com a propriedade `font-family`. Esta propriedade aceita uma lista de valores separados por vírgulas representando nomes de fontes existentes ou não no sistema do usuário. No final da lista, pode ser incluída uma referência a uma família genérica, que será usada caso nenhum dos nomes coincida com o nome de uma fonte do sistema.

A sintaxe é:

font-family: fonte1, fonte2, fonte3, ..., fonte-genérica

Exemplos:

```
H1 { font-family: garamond }
H2 { font-family: arial, helvetica, sans-serif }
H3 { font-family: courier, "courier new", monospaced }
H4 { font-family: monospaced }
```

As fontes sans-serif e monospaced são nomes genéricos. Não se referem a uma fonte em particular mas a um grupo genérico. Os outros são serif, cursive e fantasy.

Nome genérico	Serif	Sans-Serif	Monospace
Default do Unix	Times	Helvetica	Courier
Default do Mac	Times	Helvetica	Courier
Default do Win	Times New Roman	Arial	Courier New

O browser usará a primeira fonte da lista se a encontrar. Se não encontrar, irá procurar a fonte seguinte.

Se o nome de uma fonte tiver mais de uma palavra, este deverá ser colocado entre aspas. As aspas podem ser apóstrofes simples (') ou aspas duplas ("). Os apóstrofes são necessários quando for preciso especificar estilos dentro de um atributo HTML:

```
<p style="font-family: 'times new roman', sans-serif">...</p>
```

4.3.2. *font-size*

O tamanho de uma fonte é alterado usando `font-size`. Pode ser especificado em valores absolutos ou relativos. Para especificar um valor absoluto, pode-se usar:

font-size: número (pt | px | cm | mm | pc | in)

font-size: xx-small | x-small | small | medium | large | x-large | xx-large

O tamanho também pode ser especificado relativo ao elemento no qual o atual objeto está contido.

font-size: tamanho_relativo (smaller, larger)

font-size: comprimento (em ou ex)

font-size: percentagem%

Exemplos:

```
H1 { font-size: 24pt }
H1 { font-size: x-large }
H1 { font-size: smaller }
H1 { font-size: 1.5em }
H1 { font-size: 150% }
<H1 style="font-size: 1cm">
```

Os tamanhos de pontos devem ser especificados como valores inteiros (mesmo se usados cm ou in). Os browsers podem formatar os tamanhos de forma diferente e os mesmos podem ser alterados pelos usuários nos browsers atuais. As unidades válidas são: pt (pontos), px (pixels), pc (paicas), cm (centímetros), mm (milímetros) e in (polegadas).

Os tamanhos absolutos chamados pelo nome (xx-small, etc.) correspondem aos tamanhos de 1 a 7 do descritor e podem variar de acordo com a família de fontes usada (variam bastante entre plataformas também). Veja um exemplo comparativo e o resultado no Internet Explorer para Windows:

```
<style>
  p {font-family: serif}
  .t1 {font-size: xx-small}
  .t2 {font-size: x-small}
  .t3 {font-size: small}
  .t4 {font-size: medium}
  .t5 {font-size: large}
  .t6 {font-size: x-large}
  .t7 {font-size: xx-large}
</style>
(...)
<p><span class=t1>xx-small</span> |
<span class=t2>x-small</span> |
<span class=t3>small</span> |
<span class=t4>medium</span> |
<span class=t5>large</span> |
<span class=t6>x-large</span> |
<span class=t7>xx-large</span> <br>
<font size="1">size=1</font> |
<font size="2">size=2</font> |
<font size="3">size=3</font> |
<font size="4">size=4</font> |
<font size="5">size=5</font> |
<font size="6">size=6</font> |
<font size="7">size=7</font> |</p>
```

xx-small | x-small | small | **medium** | large | x-large | **xx-large**

size=1 | size=2 | size=3 | **size=4** | size=5 | size=6 | **size=7** |

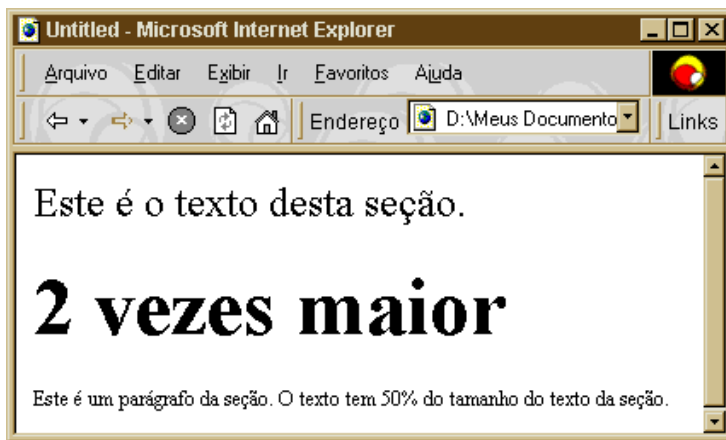
Os tamanhos relativos funcionam como o <BIG> e <SMALL>, aumentando a fonte atual por 150%. A diferença é que podem passar além do limite xx-large (ou) ou xx-small (font size=1>).

Os comprimentos referem-se a unidades comuns em tipografia. Um "em" é uma distância horizontal equivalente ao tamanho de uma fonte (se uma fonte tem 20 pontos de tamanho, um em corresponde a uma distância de 20 pixels de largura). Um "ex" é a altura das letras de caixa-baixa (sem incluir as hastes ascendentes e descendentes). Tanto "em" como "ex" usam valores relativos ao elemento que contém o elemento atual. São úteis principalmente em espaçamento, sendo pouco usados em fontes.

As porcentagens são afetadas pela herança, por exemplo:

```
<style>
  .sec {font-size: 18pt};
  .sec H1 {font-size: 200%}
  .sec P  {font-size: 50%}
</style>
</head>

<body>
<div class=sec>Este é o texto desta seção.
<h1>2 vezes maior</h1>
<p>Este é um parágrafo da seção. O texto tem 50% do tamanho do texto
da seção.</p>
</div>
```



As porcentagens de 50% e 200% são aplicadas na fonte atual, que é a fonte do bloco que contém os dois elementos (<DIV>), e que tem tamanho 18pt. O resultado é que o <H1> será exibido em tamanho 26pt e <P> em tamanho 9pt.

4.3.3. *font-style e font-weight*

O estilo de uma fonte é afetado através de duas diferentes propriedades: *font-weight*, que altera o peso da fonte, e *font-style*, que altera o estilo ou inclinação.

Sintaxe:

font-style: normal (ou italic, oblique)

Exemplos:

```
H1 { font-style: italic }
I  { font-style: normal }
<p style="font-style: oblique">...</p>
```

Sintaxe:

font-weight: normal | bold (normal=400 e bold = 700)
font-weight: bolder | lighter (valores relativos)
font-weight: 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900

Exemplos:

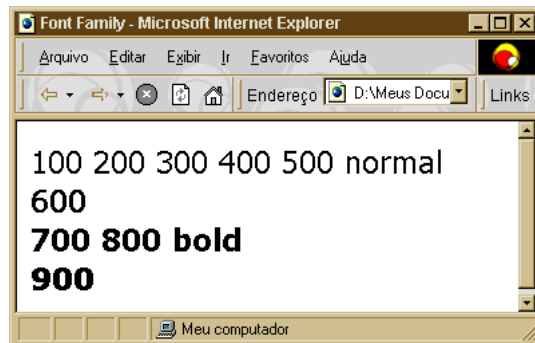
```
H1 { font-weight: normal }
B { font-weight: 900 }
<b> ... <b style="font-weight: bolder">...</b> ... </b>
```

A palavra *oblique* deve fazer com que fontes chamadas de "oblique" (se existirem no sistema) sejam usadas, assim como ocorre com fontes "italic". A rigor, *italic* é uma fonte distinta da normal, e não, meramente uma versão inclinada da mesma. Os browsers, porém, não encontrando um equivalente "italic", "oblique", "kursiv" ou similar irão inclinar o texto, simulando um itálico.

Os valores numéricos para `font-weight` oferecem um controle excepcional sobre o peso da fonte na tela, embora isto esteja limitado aos níveis disponíveis nas fontes instaladas (para um mesmo nome de fonte). Na prática, dos 9 níveis disponíveis de peso, se consegue 4 ou 5 níveis diferentes de mais pesado ou mais leve. 700 é o "bold" típico e 400 é o "normal".

O exemplo a seguir ilustra o efeito com a fonte "Tahoma" (Windows):

```
<style type=text/css>
  P      {font-family: tahoma;
           font-size: 18pt;}
  .b100 {font-weight: 100}
  .b200 {font-weight: 200}
  .b300 {font-weight: 300}
  .b400 {font-weight: 400}
  .b500 {font-weight: 500}
  .b600 {font-weight: 600}
  .b700 {font-weight: 700}
  .b800 {font-weight: 800}
  .b900 {font-weight: 900}
  .nor  {font-weight: normal}
  .bol  {font-weight: bold}
</style>
(...)
<p><span class=b100>100</span>
<span class=b200>200</span>
<span class=b300>300</span>
<span class=b400>400</span>
<span class=b500>500</span>
<span class=nor>normal</span><br>
<span class=b600>600</span><br>
<span class=b700>700</span>
<span class=b800>800</span>
<span class=bol>bold</span><br>
<span class=b900>900</span></p>
```



Os valores `lighter` e `bolder` especificam pesos de fontes relativos a algum valor herdado. Eles avançam ou retrocedem na escala de 100 a 900 relativos aos pesos de fontes.

4.3.4. *font-variant*

Atualmente a única opção disponível para esta propriedade é `small-caps`, que deve colocar o texto selecionado em maiúsculas, porém menores que as capitulares. Na prática, até as maiúsculas são reduzidas no *Internet Explorer*.

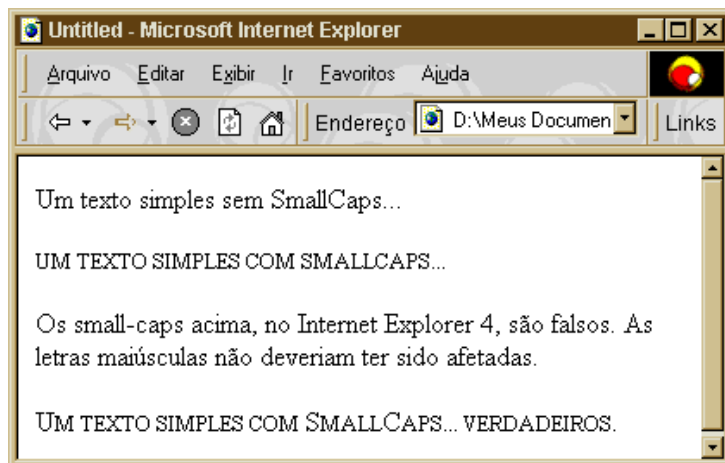
Sintaxe:

font-variant: small-caps

Exemplos:

```
<style>
  .sm {font-variant: small-caps}
  .tc {font-size: 120%}
</style>
</head>

<body>
<p>Um texto simples sem SmallCaps...</p>
<p class=sm>Um texto simples com SmallCaps...</p>
<p>Os small-caps acima, no Internet Explorer 4, são falsos. As le-
tras maiúsculas não deveriam ter sido afetadas.</p>
<p><span class=sm><span class=tc>U</span>m texto simples com <span
class=tc>S</span>mall<span class=tc>C</span>aps... verdadei-
ros</span>.</p>
```



4.3.5. *A propriedade font*

Para especificar várias propriedades de um seletor de uma vez só, pode-se usar a propriedade `font` em vez de definir em separado `font-size`, `font-weight`, `font-family`, etc. Nesta sintaxe, a ordem dos fatores é importante, porém nem todos os elementos precisam estar presentes:

```
font: font-style font-variant font-weight font-size
      line-height font-family
```

Exemplos:

```
H1 {font: italic 700 24pt Tahoma, Arial, sans-serif }
```

4.4. *Atributos de texto*

As propriedades desta seção tratam de transformações e atributos aplicados a texto, não afetando a exibição das fontes. Os atributos tipográficos afetam a forma como o texto é apresentado na tela como o espaçamento entre linhas, entre palavras, entre letras, o alinhamento de parágrafos e a endentação.

A propriedade `text-transform` permite colocar letras em maiúsculas ou minúsculas e a propriedade `text-decoration` permite acrescentar ou tirar efeitos decorativos do texto como riscados e sublinhados.

4.4.1. *text-transform*

A propriedade `text-transform` realiza transformações no formato caixa-alta ou caixa-baixa do texto. *Sintaxe:*

```
text-transform: capitalize
text-transform: uppercase
text-transform: lowercase
text-transform: none (valor default)
```

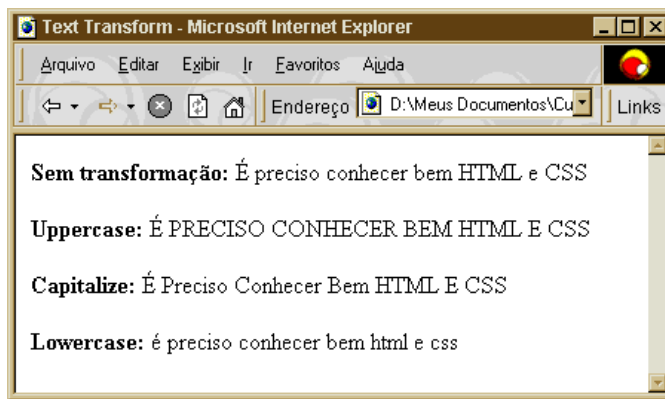
Exemplos:

```
H1 {text-transform: capitalize}
```

Capitalize coloca a primeira letra de cada palavra em maiúsculas. Uppercase coloca tudo em maiúsculas e lowercase coloca tudo em minúsculas. None remove qualquer transformação deixando o texto da forma como foi definido antes das transformações.

```
<style>
p {font-weight: bold}
span {font-weight: normal}
.non {text-transform: none}
.upp {text-transform: uppercase}
```

```
.cap {text-transform: capitalize}
.low {text-transform: lowercase}
</style>
(...)
<p>Sem transformação:
<span class=non>É preciso conhecer bem HTML e CSS</span></p>
<p>Uppercase:
<span class=upp>É preciso conhecer bem HTML e CSS</span></p>
<p>Capitalize:
<span class=cap>É preciso conhecer bem HTML e CSS</span></p>
<p>Lowercase:
<span class=low>É preciso conhecer bem HTML e CSS</span></p>
```



4.4.2. *text-decoration*

A propriedade `text-decoration` permite colocar (ou tirar) sublinhados, linhas sobre e atravessando o texto, etc. *Sintaxe:*

```
text-decoration: underline      (default em links)
text-decoration: overline
text-decoration: line-through
text-decoration: blink
text-decoration: none          (default)
```

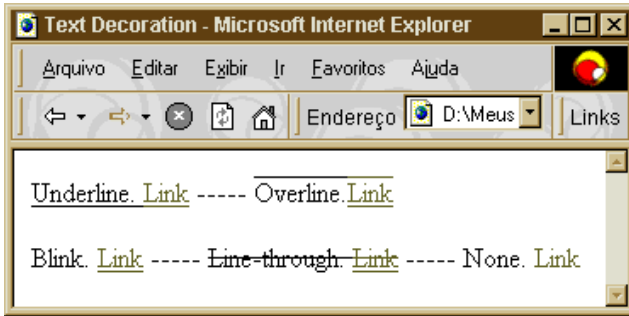
Exemplos:

```
h1 {text-decoration: overline}
<a href="algumlugar.html"
  style="text-decoration: none">Link sem sublinhado</a>

<style>
  .und {text-decoration: underline}
  .ovr {text-decoration: overline}
  .blk {text-decoration: blink}
  .lin {text-decoration: line-through}
  .non, .non a {text-decoration: none}
</style>
```

```
(...)
```

<code><p>Underline.</code>	<code>Link -----</code>
<code>Overline.</code>	<code>Link</p></code>
<code><p>Blink.</code>	<code>Link -----</code>
<code>Line-through.</code>	<code>Link -----</code>
<code>None.</code>	<code>Link</p></code>



Vínculos (links) são normalmente sublinhados na maior parte dos browsers. O sublinhado pode ser removido com a propriedade `text-decoration: none`.

O browser Netscape 4 não suporta a propriedade `overline`. O Internet Explorer não suporta a propriedade `blink`.

4.4.3. *text-align* e *vertical-align*

CSS oferece propriedades que permitem controlar o alinhamento horizontal do texto, seu alinhamento vertical e endentação do texto na primeira linha. O alinhamento horizontal é o mesmo conseguido com o atributo `align` do HTML, só que o da folha de estilos tem precedência. A sintaxe é:

```
text-align: left | right | center |
              justify
```

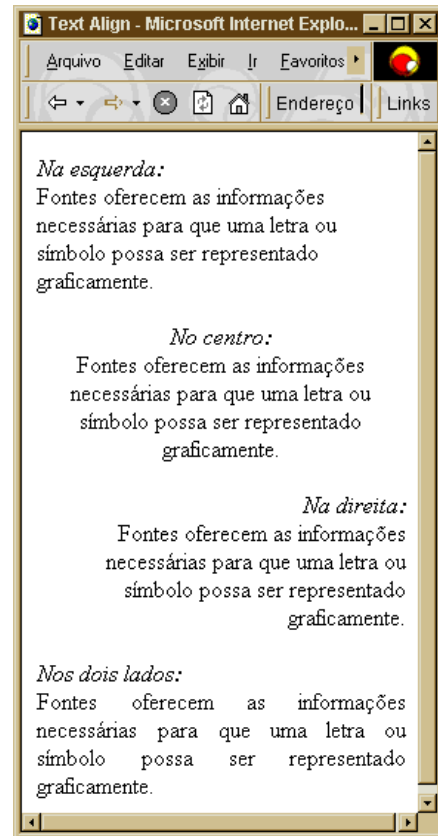
O alinhamento só se aplica a elementos de bloco (`<P>`, `<DIV>`, `H1`, etc.) e elementos como applets e imagens. A característica é herdada para sub-blocos. O valor *default* é sempre `left`. Exemplo:

```
DIV { text-align: center }
```

Alinhamento vertical em HTML só pode ser aplicado a tabelas e imagens. Com CSS, esta propriedade é estendida a qualquer elemento de texto e imagens. A sintaxe é:

```
vertical-align: baseline | top | text-top | middle |
                  bottom | text-bottom
vertical-align: sub | super
vertical-align: porcentagem %
```

O valor *default* é `baseline`. As porcentagens referem-se a altura da linha (`line-height`) do próprio elemento. Usando porcentagens negativas consegue-se sobrepor elementos.



Na prática, apenas o *Internet Explorer 4* suporta `vertical-align` com os valores `sub` e `super` (coloca elementos em subscrito ou sobrescrito).

4.4.4. *text-indent*

A propriedade `text-indent` se aplica a elementos de bloco e realiza a endentação da primeira linha. A sua sintaxe é:

text-indent: comprimento
text-indent: porcentagem

A porcentagem ocorre em relação à largura do elemento que contém o seletor. Pode ser a largura total da página, a largura da célula de uma tabela, etc. Exemplos:

```
P { text-indent: 1 cm }
P { text-indent: 50% }
<P style="text-indent: 1in">
```

A endentação tratada aqui é apenas para uma linha de texto. Para endentar blocos inteiros, deve-se usar as margens (em seção mais a frente).

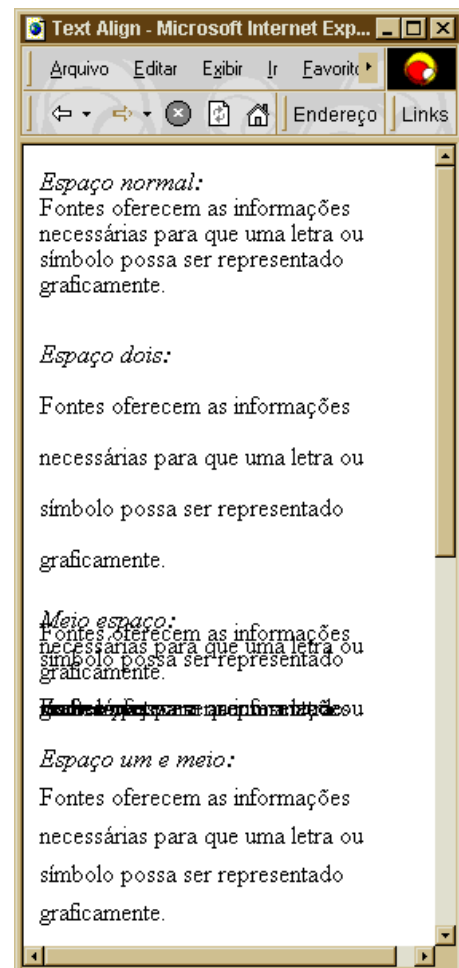
4.4.5. *line-height*

Este atributo é usado para controlar o espaço que existe antes e depois de uma linha de texto. Ela especifica a altura total de uma linha de texto. Se você tem um texto de 10 pontos e uma `line-height` de 20 pontos (`line-height: 2`), haverá 5 pontos antes e 5 pontos depois de cada linha de texto (espaço duplo). O espaço simples equivale geralmente a `line-height: 120%`. Uma `line-height` menor que o tamanho da fonte produzirá sobreposição de texto.

Embora ambos os browsers mais populares suportem este recurso, ele não ocorre da maneira correta. Os browsers não acrescentam a mesma quantidade de espaço antes e depois como esperado.

Um *bug* no *Internet Explorer 3* faz com que ele interprete valores absolutos (sem unidade) como valores em pixels. Por exemplo, 1.5 indica espaço 1 e meio ou 150%. No *Internet Explorer 3* as linhas ficam sobrepostas pois o browser interpreta a unidade como 1.5 pixels. Evite, portanto, usar valores absolutos (use porcentagens).

A sintaxe é:



line-height: número_absoluto
line-height: comprimento ou unidade
line-height: porcentagem

Exemplos:

```
H1 {line-height: 0 }      // sobreposição de linhas
H1 {line-height: 2 }     // espaço duplo
H1 {line-height: 0.3em }
H1 {line-height: 150% }  // espaço 1 e meio
```

Se você usar um valor percentual menor que 100%, um valor absoluto menor que 1 ou uma unidade menor que o tamanho da fonte, haverá sobreposição de linhas.

4.4.6. *letter-spacing*

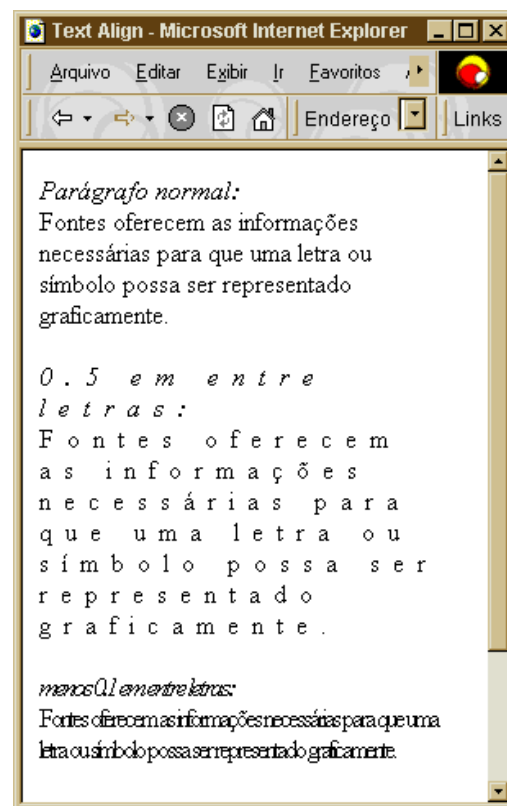
A propriedade `letter-spacing` altera o espaço entre as letras. A sua sintaxe é:

letter-spacing: normal
letter-spacing: comprimento

As unidades podem ser quaisquer uma das unidades válidas para tamanho de fontes (pt, px, pc, cm, in, mm, em e ex). Na tipografia, é mais comum usar "em" como medida de espaçamento por ser proporcional ao tamanho da fonte.

Pode-se usar também valores negativos para sobrepor caracteres, criar ligaduras (usadas em kerning) e caracteres especiais (por exemplo, sobrepondo / com \).

O suporte a `letter-spacing` nos principais browsers ainda é inconsistente. O *Netscape* (versão 4) não o suporta.



4.5. Cores

















Com as propriedades de cores, podemos controlar as cores de várias partes da página, do texto, do fundo da página e de elementos HTML. Além disso, podemos aplicar imagens de fundo em qualquer elemento, não só no elemento BODY como já se faz em HTML.

As cores definidas em CSS, assim como em HTML, podem ser especificadas por um número em hexadecimal (representando um código RGB) ou por um nome. Além dessas duas formas, podem ainda ser especificadas por três números decimais, representando também o código RGB da cor.

Os códigos RGB informam a quantidade de luz vermelha, verde e azul que compõe a cor, respectivamente. Cada cor pode ter 16 níveis de intensidade: 0 a 256 (00 a FF, em hexadecimal). O total de combinações possíveis é de 16.777.216 cores.

A exibição correta das cores depende da capacidade do vídeo onde serão vistas. Poucos sistemas têm capacidade de mostrar mais que 65.536 cores simultâneas. A maioria só mostra 256.

A tabela abaixo relaciona em **negrito** os 16 nomes padrão, suportados por todos os browsers que exibem cores, e seus respectivos códigos RGB em hexadecimal e decimal.

Cor	Nome	Cód. Decimal	Cód. Hexa	Cor	Nome	Cód. Decimal	Cód. Hexa
	red	255, 0, 0	ff0000		maroon	128, 0, 0	800000
	lime	0, 255, 0	00ff00		green	0, 128, 0	008000
	blue	0, 0, 255	0000ff		navy	0, 0, 128	000080
	yellow	255, 255, 0	ffff00		olive	128, 128, 0	808000
	aqua	0, 255, 255	00ffff		teal	0, 128, 128	008080
	fuchsia	255, 0, 255	ff00ff		purple	128, 0, 128	800080
	white	255, 255, 255	ffffff		silver	192, 192, 192	c0c0c0
	black	0, 0, 0	000000		gray	0, 0, 0	808080

Há muito mais cores com nomes suportadas pelo *Netscape* e *Internet Explorer*. Estas listadas são as únicas que fazem parte da especificação oficial do HTML 4. São todas "seguras", ou seja, fazem parte da paleta básica de 216 cores.

4.5.1. color

Define a cor do texto. A propriedade `color` substitui totalmente o descritor `` com vantagens. Pode ser aplicada localmente em um descritor (usando o atributo `style`) ou globalmente na página e no site, como qualquer outra propriedade de estilo.

A sintaxe da propriedade `color` é:

```
color: nome_de_cor
color: #número_hexadecimal
color: rgb(vermelho, verde, azul)
```

Exemplos:

```
H1 { color: green }
P { color: #fe0da4 }
EM { color: rgb (255, 127, 63) }
<EM STYLE="color: rgb (100%, 50%, 25%)">
```

Os nomes são qualquer nome válido de cor. Se o browser não encontrar o nome ao qual o estilo se refere, deve exibir a cor *default* (ou herdada). O símbolo "#" é opcional no código

hexadecimal. A intensidade da cor pode ser expressa em valores absolutos (0 a 255) ou porcentagens (0.0-100.0%).

4.5.2. *background-color*

As cores de fundo de qualquer elemento podem ser alteradas através da propriedade `background-color`. A sintaxe é:

```
background-color: transparent           (valor default)
background-color: nome_de_cor
background-color: #número_hexadecimal
background-color: rgb(vermelho, verde, azul)
```

Exemplos:

```
H1 { background-color: green }
P { background-color: #fe0da4 }
EM { background-color: rgb (255, 127, 63) }
<EM STYLE="background-color: rgb (100%, 50%, 25%) ">
```

O fundo transparente simplesmente deixa à mostra o fundo do objeto que o contém. O fundo, para texto, ocupa todo o espaço da fonte (inclusive espaço em branco acima e abaixo que fazem parte da fonte). A cor não é estendida quando o espaçamento entre linhas é aumentado em alguns browsers.

4.5.3. *background-image*

Com `background-image` é possível atribuir a qualquer elemento HTML uma imagem que será exibida no fundo, assim como as cores de fundo. A sintaxe é:

```
background-image: none           (valor default)
background-image: url(URL_da_imagem)
```

Exemplos:

```
H1 { background-image: url(http://www.xyz.com/abc.jpg) }
B { background-image: url(..../monstro.gif) navy
<TD STYLE="background-image: url(dinheiro.gif) ">...</TD>
```

As URLs são relativas à localização do arquivo que contém a folha de estilos (pode ser a própria página ou não). A cor de *backup* é usada para 'vazar' pelas partes transparentes da imagem ou prevenir contra o não carregamento do fundo (para permitir a leitura em fundo escuro pode-se usar preto como cor de *backup* de uma imagem escura).

4.5.4. *background-repeat*

CSS permite mais controle ainda sobre a imagem de fundo, facilitando a maneira como a mesma irá se repetir. A propriedade é `background-repeat`. *Sintaxe:*

```

background-repeat: repeat                (default)
background-repeat: repeat-x
background-repeat: repeat-y
background-repeat: no-repeat

```

Exemplos:

```

BODY {background-image: url(china.jpg);
      background-repeat: repeat-x }

TABLE{background-image: url(corinthians.gif)
      background-repeat: no-repeat }

```

O valor `repeat` é *default* e faz com que a imagem ocupe toda a tela. `repeat-x` faz com que a imagem seja repetida apenas horizontalmente e `repeat-y` faz com que ela seja repetida apenas verticalmente. `no-repeat` faz com que a imagem não seja repetida de forma alguma (aparecerá uma imagem apenas no canto superior esquerdo).

Para fazer a imagem aparecer em outros lugares, pode-se usar as propriedades de posicionamento do fundo da tela.

4.5.5. *background-position e background-attachment*

O posicionamento e a forma de exibição do papel de parede são controlados pelas propriedades `background-attachment` e `background-position`. A primeira define se o fundo irá ou não se mover com o texto ou ficar fixo na tela. A segunda permite o posicionamento do fundo em um local exato da tela. Infelizmente essas duas propriedades não têm suporte universal pelos browsers comerciais (apenas o *Internet Explorer* os suporta).

Sintaxe:

```

background-attachment: fixed
background-attachment: scroll

```

Exemplo:

```

BODY {background-image: url (china.jpg);
      background-attachment: fixed }

```

Sintaxe:

```

background-position: porcentagem_horiz% porcentagem_vert%
background-position: comprimento comprimento
background-position: posição_vertical posição_horizontal

```

Exemplos:

```

BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: 50% 100% }

```

```
BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: 25pt 2.5cm }
```

```
BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: center top }
```

```
BODY {background-image: url(china.jpg);
      background-repeat: no-repeat;
      background-position: left bottom }
```

Os valores de porcentagem são relativos à posição do elemento sobre o qual se aplica o estilo. As posições são sempre dadas em pares, tendo os valores separados por espaços. O primeiro valor é sempre um valor horizontal e o segundo um valor vertical. O browser coloca o bloco afetado dentro de uma "caixa invisível" e a posiciona de acordo com as porcentagens. Um valor de 100% para o primeiro valor, empurra a margem direita (oposta) desta "caixa invisível" contra a margem direita do browser.

Os valores de comprimento, assim como os de porcentagem também são dados em pares. O primeiro é a distância da margem horizontal a partir do canto superior esquerdo do objeto; o segundo é a distância da margem superior. As unidades válidas são as mesmas usadas em fontes (cm, mm, in, pc, px, pt, em, ex) e podem ser misturadas nos dois valores do par.

Os valores de posição são palavras-chave usadas também aos pares. São equivalentes das porcentagens básicas de alinhamento. O primeiro par pode ter `left` (0%), `right` (100%) ou `center` (50%). O segundo par pode ser `top` (0%), `bottom` (100%) ou `center` (50%).

4.5.6. *background*

A propriedade `background` pode ser usada para definir várias características de fundo de uma única vez. Na sintaxe abaixo, a ordem dos fatores é importante. A sua sintaxe é:

```
background: background-color background-image background-repeat
              background-attachment background-position
```

Deve haver pelo menos um valor definido, mas qualquer número de valores pode ser atribuído de uma vez.

Exemplos:

```
BODY {background: url(..duke.gif) white no-repeat fixed 50% 25%}
```

4.6. *Propriedades de classificação*

Estas propriedades classificam os elementos em categorias que podem receber estilos. Categorias podem ser listas, blocos, trechos de blocos ou itens invisíveis.

4.6.1. *display*

Esta propriedade define *como* um elemento é mostrado. A propriedade `none` desliga o elemento e fecha o espaço que o objeto antes ocupava (torna o objeto invisível). `block` abre uma nova *caixa* onde o objeto é posicionado, relativo aos outros blocos, `list-item` é um bloco com um marcador de lista e `inline` define um elemento como parte de um bloco.

Sintaxe:

display: block | inline | list-item | none

Exemplo:

```
P {display: list-item}
IMG {display: none}      // desliga todas as imagens
```

4.6.2. *white-space*

Define como o espaço em branco do elemento é gerenciado (se as linhas devem ser quebradas para que apareçam na tela ou não (`nowrap`) ou se os espaços em branco, tabulações, etc. devem ser considerados (`pre`).

white-space: normal | pre | nowrap

4.6.3. *list-style*

Esta propriedade e as propriedades `list-style-type`, `list-style-image` e `list-style-position` definem atributos para objetos de lista, como tipo de marcador, imagem do marcador e posição. Esses elementos não são suportados no *Netscape*.

list-style-type: disc | circle | square | decimal | lower-roman
upper-roman | lower-alpha | upper-alpha | none

list-style-image: url(url_da_imagem)

list-style-position: inside | outside

Exemplo:

list-style-image: url(bullet.gif)

É possível definir as três propriedades através de um atalho usando `list-type`. A ordem dos fatores é importante neste caso.

list-style: list-style-type list-style-image list-style-position

Exemplo:

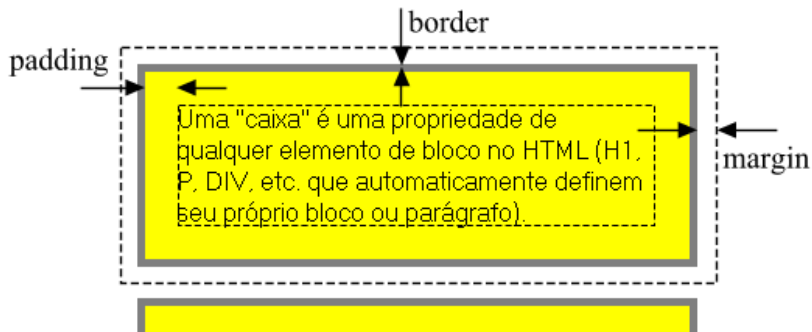
list-style: url(bullet.gif)

list-style: square outside

4.7. Controle de blocos

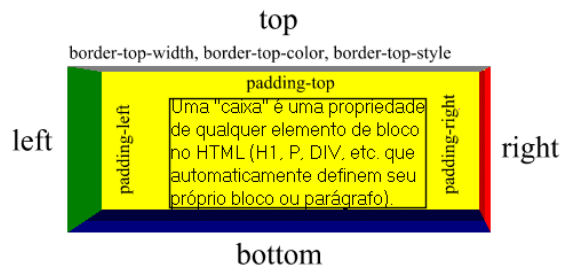
Uma “caixa” é uma propriedade de qualquer elemento de bloco no HTML (H1, P, DIV, etc. que automaticamente definem seu próprio *bloco* ou *parágrafo*). A caixa de um objeto consiste das partes seguintes:

- O elemento em si (texto, imagem)
- As margens internas do elemento (*padding*)
- A borda em torno das margens internas (*border*)
- A margem em torno da borda (*margin*)



Todo elemento de bloco tem essas propriedades. As propriedades CSS que veremos nesta seção mostrarão como alterá-las. A cor e tamanho da borda podem ser alterados assim como o fundo (como vimos na seção anterior). A margem externa é sempre transparente mas a margem interna herda a cor de fundo do objeto.

Também são alteráveis as margens internas e externas, larguras de borda, cor de borda e estilo de borda de cada um dos quatro lados de uma caixa individualmente, identificados pelos nomes *top*, *right*, *bottom* e *left*:



4.7.1. *margin e padding*

As margens externas são definidas usando a propriedade `margin` (que afeta todas as margens ao mesmo tempo) ou as propriedades `margin-top`, `margin-bottom`, `margin-right` e `margin-left` que permite alterar as margens individualmente.

Sintaxe:

```
margin-top: comprimento | porcentagem % | auto
margin-bottom: comprimento | porcentagem % | auto
margin-right: comprimento | porcentagem % | auto
margin-left: comprimento | porcentagem % | auto
```

Exemplo:

```
margin-top: 1cm; margin-left: 12pt;
```

A propriedade `margin` afeta vários aspectos das margens externas de uma vez só. A ordem dos fatores é importante. Podem ser incluídos todos quatro valores, apenas um (todas as margens iguais) ou dois (margens horizontais e verticais). *Sintaxe:*

```
margin: margin-top margin-right margin-bottom margin-left
margin: margin-top% margin-right% margin-bottom% margin-left%
margin: espaço_vertical espaço_horizontal
margin: margem_de_todos_os_lados
```

Exemplos:

```
margin: 5cm // vale para as quatro margens
margin: 5cm 2cm // 5cm margs verticais, 2cm margs horizontais
margin: 5cm 3cm 2cm 1cm // sent. horário: top, right, bottom, left
// (em cima 5; à direita 3; em baixo 2;...
```

As margens internas (*padding*) são definidas usando a propriedade `padding` (que afeta todas as margens internas ao mesmo tempo) ou as propriedades `padding-top`, `padding-bottom`, `padding-right` e `padding-left`.

Sintaxe:

```
padding-top: comprimento | porcentagem %
padding-bottom: comprimento | porcentagem %
padding-right: comprimento | porcentagem %
padding-left: comprimento | porcentagem %
```

A propriedade `padding` afeta vários aspectos das margens internas de uma vez só. A ordem dos fatores é importante. Podem ser incluídos todos quatro valores ou apenas um. *Sintaxe:*

```
padding: padding-top padding-right padding-bottom padding-left
padding: padding-top% padding-right% padding-bottom% padding-left%
```

```
padding: espaço_vertical    espaço_horizontal
padding: margem_de_todos_os_lados
```

4.7.2. *border-width*

Pode se controlar vários aspectos das bordas como a sua espessura em cada um dos quatro lados, suas cores (também cada um dos quatro lados) e estilos (idem). Isto pode ser feito de diversas maneiras. Para que as bordas apareçam é preciso primeiro que o estilo (`border-style`) seja definido. Por exemplo:

```
border-style: solid
```

A espessura das bordas pode ser controlada através da propriedade `border-width`, afetando as espessuras de todos os lados da borda, ou individualmente através de `border-top-width`, `border-bottom-width`, `border-right-width` e `border-left-width`. *Sintaxe:*

```
border-top-width: comprimento | thin | medium | thick
border-bottom-width: comprimento | thin | medium | thick
border-right-width: comprimento | thin | medium | thick
border-left-width: comprimento | thin | medium | thick
```

Exemplos:

```
border-bottom-width: thick; border-right-width: 5.5px;
border-left-width: 0.2cm
```

As propriedades das bordas podem ser tratadas em grupo, com `border-width`. A ordem dos fatores é importante. Podem ser incluídos todos os quatro valores, dois (referindo-se às bordas horizontais e verticais) ou apenas um (afetando todas as bordas). *Sintaxe:*

```
border-width: border-top-width    border-right-width
                border-bottom-width border-left-width
```

Exemplos:

```
border-width: 5cm           // vale para as quatro bordas
border-width: 5cm 2cm      // 5cm verticais, 2cm horizontais
border-width: 5cm 3cm 2cm 1cm // horário: top, right, bottom, left
```

4.7.3. *border-color*

A propriedade `border-color` é um atalho que permite que se altere a cor de uma ou de todas as quatro bordas ao redor de um elemento que também podem ser definidas individualmente através de `border-top-color`, `border-bottom-color`, `border-right-color` e `border-left-color`.

```
border-top-color: cor (nome ou código)
border-bottom-color: cor (nome ou código)
```


border-right-color: cor (nome ou código)

border-left-color: cor (nome ou código)

Exemplos:

border-bottom-color: rgb(231,45,112); **border-right-color:** 0fa97b;

border-left-color: navy

As propriedades das bordas podem ser tratadas em grupo, com `border-color`. A ordem dos fatores é importante. Podem ser incluídos todos os quatro valores, dois (referindo-se às bordas horizontais e verticais) ou apenas um (afetando todas as bordas). *Sintaxe:*

border-color: border-top-color border-right-color
 border-bottom-color border-left-color

Cada um dos `border-xxx-color` acima é uma cor (RGB, hexadecimal ou nome). Pode-se alterar todas as bordas de uma vez, apenas as duas verticais e horizontais ou as quatro individualmente.

Exemplos:

border-color: red // red para as quatro bordas

border-color: rgb(255, 0, 0) // red

border-color: rgb(100%, 0, 0) // red

border-color: red 0000ff // red verticais, 0000ff horizontais

border-color: red blue yellow cyan // 4 cores sentido horário

4.7.4. *border-style*

O estilo de cada uma das quatro bordas pode ser alterado com `border-style`. Também é possível defini-los individualmente usando `border-top-style`, `border-bottom-style`, `border-right-style` e `border-left-style`. São vários os estilos disponíveis (tracejado, pontilhado, várias versões de 3D, etc.).

border-top-style: none | dotted | dashed | solid | double |
 groove | ridge | inset | outset

border-bottom-style: nome de estilo (um dos nomes acima)

border-right-style: nome de estilo

border-left-style: nome de estilo

Exemplos:

border-bottom-style: none **border-right-style:** solid;

border-left-style: inset

As propriedades das bordas podem ser tratadas em grupo, com `border-style`. A ordem dos fatores é importante. Podem ser incluídos todos os quatro valores, dois (referindo-se às bordas horizontais e verticais) ou apenas um (afetando todas as bordas). *Sintaxe:*

border-style: border-top-style border-right-style
 border-bottom-style border-left-style

Exemplos:

```
border-style: solid none inset outset;
border-style: solid
border-style: inset outset
```

Cada um dos `border-xxx-style` acima é um dos seguintes valores: `none`, `dotted`, `dashed`, `solid`, `double`, `groove`, `ridge`, `inset`, `outset`. No *Netscape* e *Internet Explorer*, funcionam os estilos `solid`, `inset` (efeito baixo-relevo) e `outset` (efeito alto-relevo). No *Netscape* `solid` é *default* mas no *Explorer*, o *default* é `none`, portanto, uma borda não aparece se a propriedade `border-style` não for definida antes.

4.7.5. *border*

As propriedades `border-top`, `border-bottom`, `border-left` e `border-right` agrupam as propriedades de cor, estilo e espessura para cada uma das quatro bordas.

```
border-top:   border-width border-style border-color
border-bottom: border-width border-style border-color
border-left:  border-width border-style border-color
border-right: border-width border-style border-color
```

A propriedade `border` é uma forma reduzida de definir tudo isto de uma vez só para todas as bordas e de forma idêntica (não é possível especificar valores diferentes para as bordas neste caso). Todos os itens não precisam aparecer, mas a ordem dos fatores é importante:

```
border:       border-width border-style border-color
```

4.7.6. *width e height*

As propriedades `width` e `height` modificam a altura e largura de um bloco da mesma forma que atributos HTML `width` e `height` usados em imagens e applets no HTML. Com folhas de estilos podem ser usados para redimensionar a "caixa" de qualquer elemento de bloco. *Sintaxe:*

```
width: comprimento | auto
height: comprimento | auto
```

4.7.7. *float*

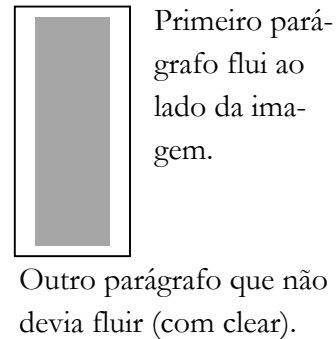
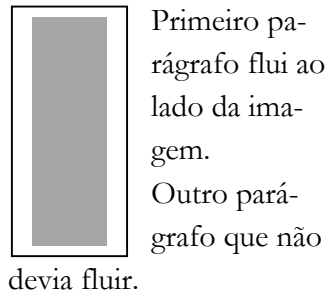
A propriedade `float` permite que um bloco qualquer seja posicionado à direita ou esquerda da janela do browser, fazendo com que o texto restante flua em sua volta. Permite que qualquer bloco se comporte como uma imagem que faz uso dos atributos `align=left` e `align=right` no HTML. *Sintaxe:*

```
float: left | right | none
```

4.7.8. *clear*

E finalmente, para evitar que um bloco flua em torno de uma imagem ou bloco que utiliza a propriedade `float`, existe a propriedade `clear`, que deve ser atribuída a um bloco ou parágrafo que é afetado pela flutuação de um bloco. Faz a mesma coisa que o atributo `clear`, usado no `
` em HTML, só que aqui ela é suportada em qualquer elemento (bloco ou não).

clear: none | left | right | both



4.8. *Posicionamento*

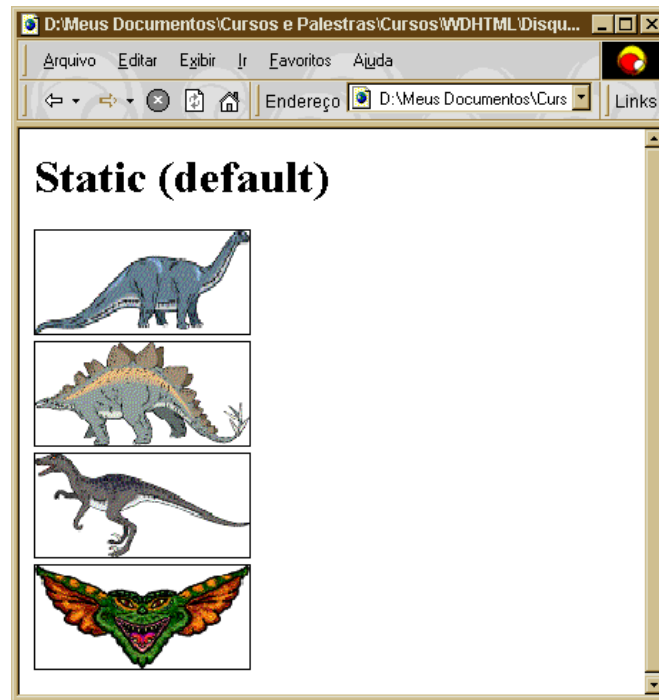
O posicionamento de objetos em HTML pode ser conseguido em termos absolutos ou relativos usando CSS 2. As propriedades permitem o posicionamento em três dimensões (horizontal, vertical e em camadas). Embora os recursos de posicionamento não façam parte do CSS1, tanto o *Netscape Navigator 4* como o *Internet Explorer 4* o suportam.

4.8.1. *position, top e left*

Esta propriedade precisa de mais duas propriedades que definem o posicionamento de um objeto. Os atributos localizam o objeto na tela de forma *absoluta* ou de forma *relativa*. A origem da posição absoluta será a posição do objeto no nível imediatamente superior. O posicionamento relativo se refere à posição anterior do objeto. *Sintaxe:*

position: absolute | relative
left: comprimento | porcentagem | auto
top: comprimento | porcentagem | auto

Exemplo: considere as quatro imagens a seguir:



```
<div class="imagem1">
 1 Posição 0 0
</div>

<div class="imagem2">
 2 Posição 0 200
</div>

<div class="imagem3">
 3 Posição -25 -25
</div>

<div class="imagem4">
 4 Posição 100 0
</div>
```

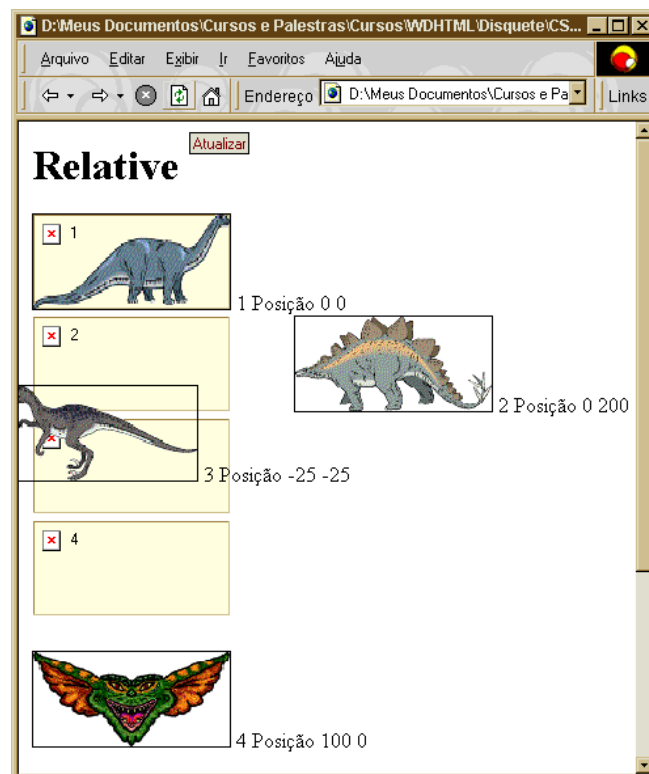
Aplicando a seguinte folha de estilos para posicionar as quatro imagens relativamente à sua localização original obtemos a imagem ao lado. Os retângulos claros indicam a posição original da imagem:

```
DIV.imagem1 {
    position: relative;
    top: 0px;
    left: 0px;
}
```

```

DIV.imagem2 {
  position: relative;
  top: 0px;
  left: 200px;
}
DIV.imagem3 {
  position: relative;
  top: -25px;
  left: -25px;
}
DIV.imagem4 {
  position: relative;
  top: 100px;
  left: 0px;
}

```

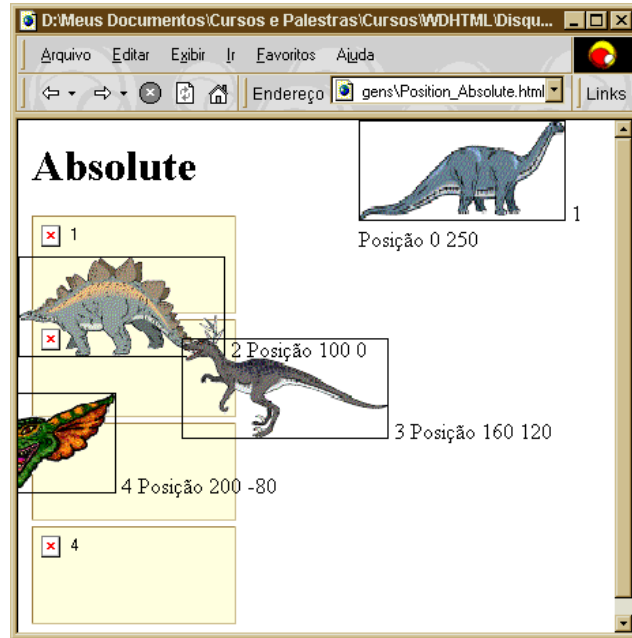


Usando posicionamento absoluto, o bloco é movido em relação ao canto superior esquerdo do browser.

```

DIV.imagem1 {
    position: absolute;
    top: 0px;
    left: 250px;
}
DIV.imagem2 {
    position: absolute;
    top: 100px;
    left: 0px;
}
DIV.imagem3 {
    position: absolute;
    top: 160px;
    left: 120px;
}
DIV.imagem4 {
    position: absolute;
    top: 200px;
    left: -80px;
}

```



4.8.2. *z-index*

A propriedade `z-index` permite ordenar os objetos em camadas. É um eixo de profundidade. Para usá-la, basta definir em cada objeto:

z-index: número

onde número é a camada de exibição. Quanto maior o número, mais alta a camada. o corresponde à camada do texto. Se um objeto tiver `z-index` menor que zero aparecerá atrás do texto. Se `z-index` for maior que zero, aparecerá na frente. Quando não é definido ou quando `z-index: 0` o bloco ocupará a mesma camada que o texto.

4.8.3. *visibility*

Esta propriedade serve para tornar um bloco visível ou invisível. Difere de `display` porque não remove o espaço antes ocupado pela imagem:

visibility: hidden | visible

Exemplo:

```
IMG {visibility: hidden} // torna invisíveis todas as imagens
```

4.9. Exercícios

4.9.1. Testes sobre Folhas de Estilo

- Qual das seguintes regras de estilo está incorreta? Marque uma.
 - `a:link {color: rgb(0%,40%,40%)}`
 - `div.code pre {margin-bottom: 0px}`
 - `body {font-size: 0.5cm, color: yellow, background: black}`
 - `.botcor {font-size: 16pt; font-family: tahoma, sans-serif;}`
 - Estão todas corretas.
- Qual dos seguintes trechos de código é ilegal dentro de um arquivo `.css`? Marque uma.
 - `span.value {color: maroon}`
 - `/* <H1>Titulo</H1> */`
 - `@import url(http://ww.estilos.org/estilo.css);`
 - `<STYLE>`
 - Nenhuma das alternativas é ilegal dentro de um arquivo CSS.
- Qual das regras abaixo, de uma folha de estilos, declara que os parágrafos e células de dados de tabelas terão texto vermelho?
 - `P TD {color: red}`
 - `P: TD {color: ff0000}`
 - `P, TD {color: rgb(100%, 0%, 0%)}`
 - `P; TD {color: rgb(255, 0, 0)}`
 - `P, TD {color=red}`
- Qual das declarações abaixo, contida em uma página HTML, a vincula à folha de estilos `basico.css`, localizada no mesmo diretório que a página?
 - `<LINK REL=StyleSheet HREF="basico.css">`
 - `<LINK REL=StyleSheet SRC="basico.css">`
 - `Folha de estilos`
 - `<FRAME SRC="basico.css" REL="StyleSheet">`
 - `Folha de estilos`
- Considere o seguinte trecho de código HTML:


```
<div>
<p>Parágrafo</p>
</div>
```

Quais declarações abaixo, em um bloco `<STYLE>` do arquivo que contém o trecho acima, farão com que o texto do parágrafo tenha tamanho 10pt em um browser que suporte folhas de estilo? Marque uma ou mais.

- `div {font-size: 20pt}`
`p {font-size: 50%}`
- `div {font-size: 10pt}`

- c) `p {font-size: 10pt}`
- d) `div {font-size: 5pt}`
`p {font-size: 100%}`
- e) `p div {font-size: 10pt}`

6. Considere a seguinte folha de estilos, com uma única regra, vinculada a uma página HTML.

```
P {color: green}
```

Dentro dessa página, logo depois da instrução que vincula o estilo à página, há um bloco `<STYLE>`, com a seguinte regra:

```
P {color: red}
```

A página possui dez parágrafos. Um deles atribui um estilo local usando o atributo `STYLE`, da forma:

```
<P STYLE="color: blue">Parágrafo</P>
```

Supondo que a página seja visualizada em um browser que suporte folhas de estilo CSS, qual é a cor da maior parte dos parágrafos dessa página?

- a) azul (blue)
- b) vermelha (red)
- c) verde (green)
- d) preta (black)
- e) indefinida

7. Identifique as alternativas que contém HTML ou CSS incorretos:

- a) `Texto`
- b) `Texto`
- c) `<div class="sec1">Tem mais texto.</div>`
- d) `Itens e <div class="sec1">seções</div> especiais.`
- e) `<div style="P {color: yellow}"><P>Texto amarelo</P></div>`

8. Considere o código HTML abaixo:

```
<div class=sec2>
  <p class=novo>Texto modificado</p>
</div>
```

Quais das regras de estilo abaixo fará com que o parágrafo seja exibido na cor azul, em um browser que suporte folhas de estilos CSS?

- a) `P {color: blue}`
- b) `.sec2 {color: blue}`
- c) `P.novo {color: blue}`
- d) `.sec2 .novo {color: blue}`
- e) `P.sec2 {color: blue}`

9. Qual das regras abaixo retira o sublinhado apenas dos *links* visitados? Marque uma.

- a) `a: visited {text-decoration: none}`
- b) `a, visited {text-decoration: none}`

- c) `a.visited {text-decoration: none}`
 - d) `a visited {text-decoration: none}`
 - e) Nenhuma das regras anteriores.
10. Marque apenas as alternativas verdadeiras
- a) Uma mesma folha de estilos pode ser usada por várias páginas.
 - b) Uma mesma página pode usar várias folhas de estilo.
 - c) Se um browser não suportar uma folha de estilos requerida pela página, poderá haver uma degradação na qualidade da apresentação mas nunca haverá perda de informação.
 - d) É possível construir um site inteiro usando apenas CSS.
 - e) A linguagem CSS usada para construir folhas de estilo é uma recomendação do W3C – consórcio de empresas que estabelece os padrões para a Web.

4.9.2. Exercícios com Folhas de Estilo

Os exercícios a seguir têm a finalidade de explorar as principais propriedades do CSS e permitir que se verifique o suporte a elas nos browsers populares. Eles são mais didáticos do que úteis. O objetivo é apenas praticar com folhas de estilos. Para realizá-los, use os arquivos disponíveis no CD do ASIT.

Conceitos básicos

1. Crie uma folha de estilos, chame-a de `basico.css`, e a carregue no arquivo `StyleTest.html`.
2. Nesta folha de estilos, usando o mínimo de declarações possível, declare:
 - a) que todo H1 tenha fonte Tahoma, ou sans-serif, se Tahoma não existir
 - b) que todo o texto do corpo (BODY) do arquivo tenha tamanho 10 pontos
 - c) que todos os H1, H2 e H3 sejam vermelhos
 - d) que os H1 tenham tamanho 24 pontos
 - e) que os H2 tenham tamanho 18 pontos
 - f) que os H3 tenham tamanho 14 pontos
3. Mude a cor do fundo da página para azul marinho (navy) e a cor *default* do texto para branco em uma única declaração.
4. Faça com que todo texto marcado em itálico apareça em azul ciano (cyan).

Formas de usar CSS

5. Carregue a folha de estilos `basico.css` em outros arquivos HTML e veja o que acontece. Faça com que uma dessas outras páginas tenha uma cor de fundo clara (amarela, por exemplo) e cor de texto escuro (diferente daquela definida por `basico.css`) sem que isto altere as outras páginas que usam o mesmo arquivo.
6. Faça com que o primeiro parágrafo do arquivo `StyleTest.html` tenha texto verde.

7. Faça com que a célula do meio da tabela de StyleTest.html tenha texto vermelho sobre fundo amarelo (a tabela 3x3 encontra-se no meio da página).

Classes, links, seletores de contexto

Para os exercícios abaixo, desligue a folha de estilos usada nos exercícios anteriores (mude o nome ou remova o elemento <LINK>) para que a página fique limpa outra vez. Use uma nova folha de estilos para aplicar as alterações a seguir.

8. Defina classes sec2, sec3, sec31 e sec32 para as seções (<DIV>) do documento StyleTest.html. As seções estão indicadas em comentários HTML (por exemplo: <!--Seção 2 -->). Aplique um fundo diferente (imagem ou cor) nessas seções para diferenciá-las das outras.
9. Tire os sublinhados de todos os links e substitua-os por um fundo cinza claro.
10. Faça com que o link ativo (active) fique em negrito; que o link normal tenha tamanho 10pt e que mostre fundo amarelo quando o mouse estiver sobre ele (hover); e que o link visitado não tenha mais cor de fundo mas recupere o sublinhado. Obs: Para fazer um link ainda não visitado, faça um link para qualquer recurso do sistema de arquivos; para ver o link ativo, clique no link e segure o mouse.
11. Faça com que:
 - a) todos os itálicos dentro de negritos sejam colocados em maiúsculas (use text-transform: uppercase).
 - b) todos os negritos dentro de itálicos sejam sublinhados
 - c) todos os negritos que estejam dentro de um bloco itálico que está dentro de um bloco LI que está dentro de uma UL que está em outra UL, sejam colocados em fonte arial, em maiúsculas e em vermelho.

Fontes

Crie uma nova folha de estilos (fontes.css) para aplicar fontes. Vincule (LINK) ou importe-a (@import) em seus arquivos.

12. a) Aplique Verdana como fonte *default* em todo o site. Garanta que, se Verdana não existir, Arial será usada, e se esta não existir, será usada a *default* sans-serif. Para testar, mude os nomes das primeiras fontes para nomes desconhecidos do sistema. b) Teste a compatibilidade dos dois browsers em relação ao suporte de fontes com nomes longos (entre aspas) em folhas de estilo locais e remotas.
13. Faça com que os de seus parágrafos sejam 20% maiores que o texto normal destes parágrafos.

Atributos de texto e classificação

Crie uma nova folha de estilos para esses exercícios.

14. a) Aplique um espaçamento de 1 cm entre palavras de um parágrafo seu texto (isto poderá não funcionar devido à falta de suporte dos browsers). b) Aplique um espaçamento de 1 cm entre as letras de outro parágrafo. Teste nos dois browsers.
15. Defina todos os títulos H2 como sendo caixa alta (uppercase).
16. Experimente com as propriedades text-decoration (use overline e outras propriedades em blocos criados para mostrar cada propriedade).
17. Elimine o espaçamento entre os parágrafos (<P>) usando {margin-top: 0pt}, endente a primeira linha e coloque todos os seus parágrafos, com exceção dos parágrafos da terceira seção, com alinhamento justificado. O alinhamento deve ser aplicado apenas nos parágrafos e não em listas ou tabelas.

Cores

18. Experimente com cores, aplicando cores em textos, backgrounds de diversos componentes da página, inclusive formulários (<INPUT> e <SELECT>). Use as três formas (url(r, g, b), rrggbb e nomes) e veja como ocorre o suporte dos browsers em folhas de estilo locais e externas. Dica: crie uma folha de estilos só para este exercício.

Fundos, Imagens e Repetições

19. Inclua a imagem rabbit.gif (ou outra qualquer do subdirectório 3_Imagens do CD do A-SIT) no fundo da página StyleTest.html (usando uma nova folha de estilos: background.css). Experimente com posicionamento, fazendo a imagem ficar fixa em vez de rolar na tela. Teste nos dois browsers. Experimente com repetições, fazendo a imagem repetir na vertical, na horizontal e não repetir. Veja o que acontece nos dois browsers.
20. Numa outra folha de estilos (para este exercício), posicione a imagem no centro da página, sem repetições e uma outra imagem no centro da tabela, também sem repetições.
21. Posicione (outra folha de estilos) o rabbit.gif (ou outra imagem) em uma posição a 4cm da margem esquerda e a 7cm da margem superior. Na seção 2 (sec2), posicione bart.gif, repetindo somente na vertical, afastado 11cm da margem esquerda e iniciando 1 cm abaixo da margem superior.

Posicionamento e Layout

22. Remova o espaço entre todos os parágrafos de StyleTest.html. Aplique um text-indent de 1cm em cada parágrafo.
23. Faça com que os blocos (parágrafos e cabeçalhos) da seção 3.1 e 3.2 (DIV) tenham 0,3 cm de margem esquerda e direita, e 0,5cm de margem superior e inferior, em relação às bordas da seção.

24. Faça com que as seções 3.2 e 3.1 tenham uma margem externa de 0,5 cm em relação à seção 3.
25. Aplique uma borda azul, sólida, de 3mm acima da seção 2 (<DIV>), uma outra, também de 3mm, abaixo, na cor amarelo. Dos lados, coloque bordas vermelhas, de 5mm.
26. Aplique uma borda verde, de 4mm à esquerda de todos os parágrafos da seção 2. Entre a borda e o texto deve haver um espaço de 5mm. Entre a borda e a margem da página, mais 5mm (sem levar em conta o offset).
27. Sem usar tabelas, aplique uma largura máxima de 500 pixels em toda a página.
28. Faça com que a seção 3.1 tenha largura máxima de 220 pixels e flutue para a direita, deixando o restante do texto fluir em volta dela.

Posicionamento absoluto

29. Monte o quebra-cabeças do exercício 1, do livro 3 (Além do HTML) sem usar tabelas (usando apenas posicionamento de folhas de estilo).
30. Diagrame a página do exercício 2, do livro 3, sem usar tabelas (usando apenas posicionamento de folhas de estilo).