



**Grupo de Bolsistas Instrutores
do Departamento de Informática**

Construção de Sites I

Módulo EAD



Html, CSS e
JavaScript

INDICE

• Funcionamento de aplicações Web	5
○ O que é HTML?	5
○ Navegadores	5
○ Cliente/Servidor	6
• Linguagem HTML	7
○ Tags	7
○ Elementos HTML	8
○ Atributos HTML	8
○ Definindo a estrutura da página HTML	8
○ Formatação de texto	8
○ Alinhamento ao centro	9
○ Alinhamento à direita	10
○ Alinhamento à esquerda	10
○ Negrito	11
○ Itálico	11
○ Sublinhado	12
○ Riscado	12
○ Parágrafo	13
○ Quebra de linha	13
○ Letra pequena	14
○ Letra grande	14
○ Links	15
○ Cor da fonte	15
○ Tipo da fonte	16
○ Tamanho da fonte	16
○ Marcadores	17
○ Marcadores com parâmetro <i>type</i>	17
○ Barra horizontal	19
○ Cor de fundo	19
○ Criação de tabelas	20
○ Imagem como plano de fundo	21
○ Imagens	22
○ Divisões de layout	23

○ Aprendendo mais sobre tabelas	25
○ Principais atributos de uma tabela	26
○ Principais atributos de uma célula	28
○ Linhas de tabelas	28
● CSS – Folhas de Estilo	30
○ O que são folhas de estilo?	30
○ Para que servem as folhas de estilo?	31
○ Regras, declarações e seletores	33
○ Múltiplas declarações e seletores	34
○ Comentários e instruções	35
○ Valores	36
○ Herança	37
○ Descritores HTML especiais	38
○ Como incluo estilos em uma página?	38
○ Classes e ID's	41
○ Links	42
○ Seletores de texto	43
○ Cascata de folhas de estilo	43
○ <i>Styling Background</i>	45
○ <i>Styling Text</i>	47
○ <i>Styling Fonts</i>	49
○ <i>Styling Links</i>	51
○ <i>Styling Lists</i>	51
○ <i>Styling Tables</i>	52
○ Box model	54
● JavaScript	61
○ Incorporando o JavaScript em HTML	61
○ Orientação à objetos	63
○ Manipulação de objetos	63
○ Propriedades de objetos	63
● Fontes e referências	68
● Exercícios	69

- Funcionamento de aplicações Web –

• O que é HTML?

HTML ou HyperText Markup Language, é uma das linguagens utilizadas para desenvolver páginas na internet, existem outras linguagens mais avançadas, porém dificilmente você verá um site que não utilize HTML.

O HTML possibilita apresentar informações (documentação de pesquisas científicas) na Internet. Aquilo que você vê quando abre uma página na Internet é a interpretação que seu navegador faz do HTML. A linguagem é baseada em “marcas” ou *tags* e sua forma de programação é bem simples, pode-se criar sua página digitando um “programa” num editor de texto qualquer. Alguns exemplos básicos de tags, que serão vistas com mais detalhes posteriormente são: `<html>`, `<head>`, `<body>`, `<title>`...

Curiosidade: Você pode ver o código-fonte das páginas usando o seu navegador para conhecer a linguagem (algumas páginas não usam só o HTML).

Saiba como ver o código-fonte nos navegadores mais usados:

Mozilla Firefox - clique com o botão direito na página e em seguida “código-fonte”;

Google Chrome - clique com o botão direito na página e em seguida “exibir código de fonte da página”;

Internet Explorer - clique em “exibir e depois “código-fonte”;

• Navegadores

Um navegador, também conhecido pelos termos ingleses Web browser ou simplesmente browser, é um programa de computador que habilita seus usuários a interagirem com documentos virtuais da internet (essa interação é feita através de protocolos, visto no próximo tópico), também conhecidos como páginas da web que podem ser escritas em linguagens como HTML, ASP, PHP, com ou sem folhas de estilos em linguagens como o CSS e que estão hospedadas num servidor Web.

No nosso caso usaremos HTML para o desenvolvimento e CSS para a formatação.

- **Cliente/Servidor**

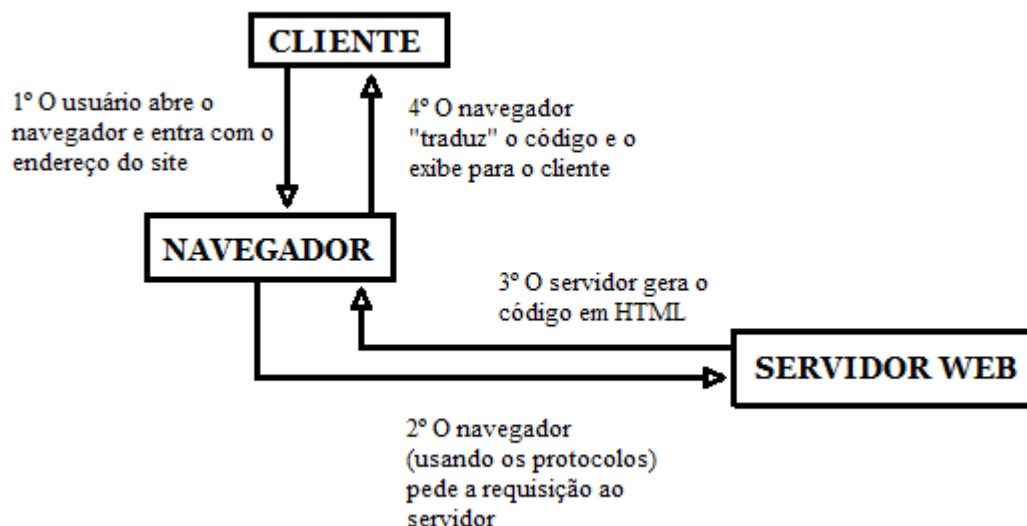
As informações disponíveis na Web necessitam de um pedido de serviço de transferência de arquivos (construídos com HTML) pelo cliente, e para isso é necessário um navegador.

Antes de acessar o site, é necessário hospedá-lo em algum servidor e para isso é preciso ter um registro de um domínio (endereço) pois será através deste endereço que o seu site será acessado na internet. Uma vez escolhido o nome e a extensão será preciso fazer a compra e o pagamento da taxa de uso do domínio entretanto também existem servidores de hospedagem gratuitos.

Depois de hospedado, com o navegador (requisitado pelo cliente) pode-se acessar a página, e para isso o navegador precisa “falar” a mesma língua do servidor usando protocolos de comunicação.

Para cada tipo de serviço, um protocolo diferente é utilizado, por exemplo, para que o navegador possa requisitar ao servidor um documento html, antes ele deve estabelecer uma comunicação com o mesmo, o que é feito através dos protocolos TCP (Transmission Control Protocol) IP (Internet Protocol) e precisamos também de um protocolo que coordene a requisição e transmissão do documento html requerido, para esse caso o protocolo utilizado é o HTTP (Hyper Text Transfer Protocol).

Assim, utilizando o HTTP, TCP,IP e outros protocolos como FTP, SMTP e POP (Esses para envio e recepção de arquivos e envio e recepção de e-mails), o navegador Web consegue, a partir de uma URL (Uniform Resource Locator), acessar seu conteúdo no servidor, interpretar e traduzir o código html a fim de se obter um documento contendo texto, imagens, sons, vídeos, etc. O esquema abaixo mostra como isso funciona.



- Linguagem HTML -

- Tags

São necessárias tags para que o navegador possa descrever como deve ser apresentado o website.

As tags tem sempre o mesmo formato começam com o sinal '<' e terminam com '>'.

Exemplo:

`<html>`

Há dois tipos de tags, as de abertura e de fechamento. Tudo que estiver contido entre uma tag de abertura e uma tag de fechamento será processado segundo o comando contido na tag.

Exemplo:

abertura: `<html>`

fechamento: `</html>`



As tags básicas de HTML são :

`<html>`: define o início de um documento HTML e indica ao navegador que todo conteúdo posterior deve ser tratado como uma série de códigos HTML.

`<head>`: define o cabeçalho de um documento HTML, que traz informações sobre o documento que está sendo aberto.

`<body>`: define o conteúdo principal, o corpo do documento. Esta é a parte do documento HTML que é exibida no navegador. No corpo pode-se definir propriedades comuns a toda página, como cor de fundo, margens, e outras formatações.

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

  <body>
  </body>

</html>
```

- **Elementos HTML**

Basicamente tudo o que estiver entre duas tags de mesmo nome é um elemento. Por exemplo, o elemento BODY é tudo o que se encontra entre `<body>` (tag de abertura) e `</body>` (tag de fechamento). Outros exemplos de elementos são os cabeçalhos, os parágrafos e as tabelas.

- **Atributos HTML**

São características dos elementos que podem ser modificadas pelo desenvolvedor. As propriedades da fonte, por exemplo, são atributos que podem ser modificados no cabeçalho, nos parágrafos, etc. Os atributos devem ser sempre modificados na tag de abertura.

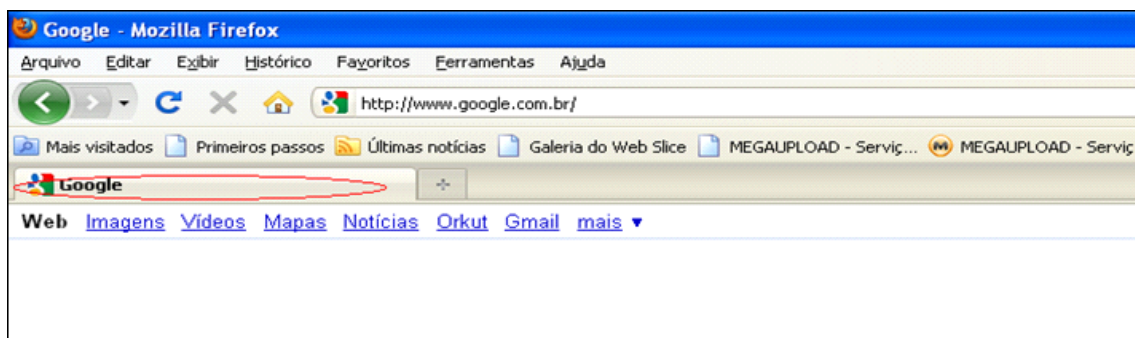
Exemplo:

```
<body atributo=valor>  
    conteúdo  
</body>
```



- **Definindo a estrutura da página em HTML**

Cabeçalho (`<head>`): dentro dele utilizamos a tag `<title>` que define o título da página, que é exibido na barra de título dos navegadores:



Corpo (`<body>`): no corpo encontramos as tags de formatação, de tabelas e imagens, etc.

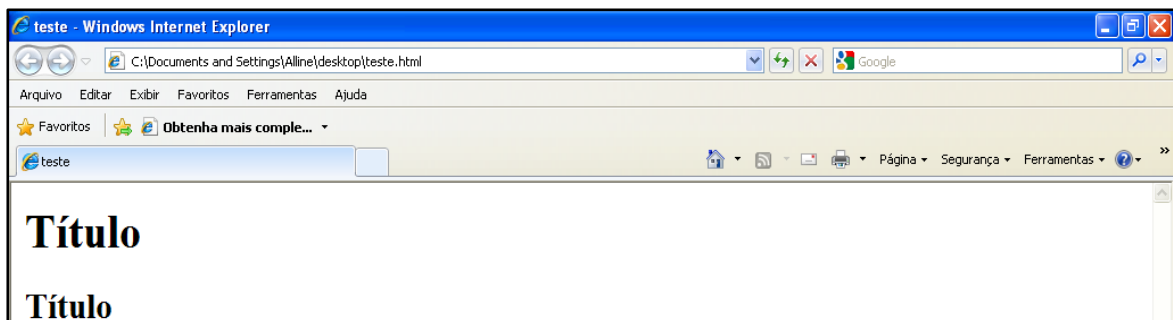
- **Formatação de texto**

`<h1...h6>`: As tags `<h1>`, `<h2>`, `<h3>`, `<h4>`, `<h5>` e `<h6>` informam ao navegador que trata-se de um cabeçalho (h vem de "heading" - cabeçalho), sendo `<h1>` o cabeçalho de primeiro nível e aquele apresentado com o maior

tamanho de texto, `<h2>` o cabeçalho de segundo nível e aquele apresentado com tamanho de texto um pouco menor e `<h6>` o cabeçalho de sexto nível e aquele apresentado com o menor tamanho de texto.

Exemplo:

```
<html>
<head>
    <title> Teste </title>
</head>
<body>
    <h1> Título </h1>
    <h2> Título 2 </h2>
</body>
</html>
```



- Alinhamento ao centro (centralizar)

Exemplo:

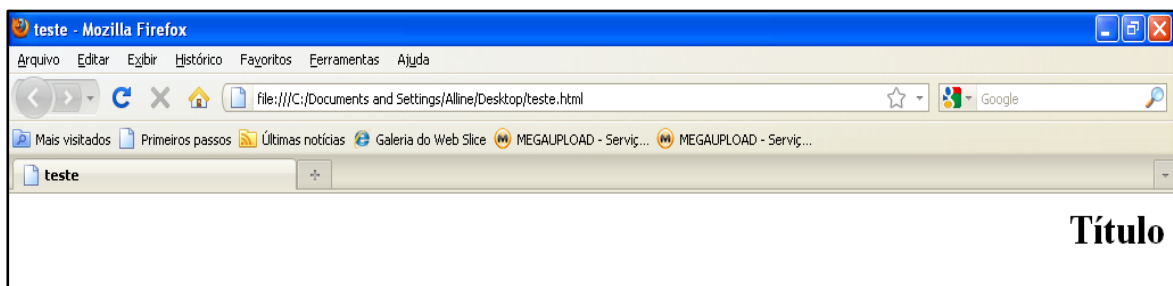
```
<html>
<head>
    <title> Teste </title>
</head>
<body>
    <h1 align=center> Título </h1>
</body>
</html>
```



- Alinhado à direita

Exemplo:

```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <h1 align=right > Título </h1>
</body>
</html>
```

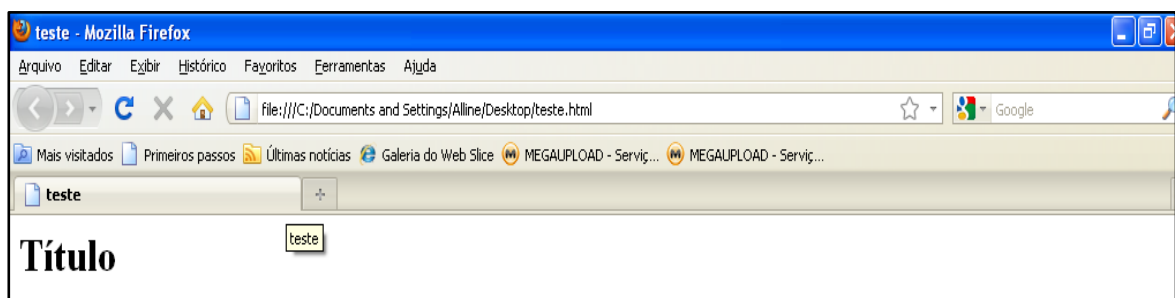


- Alinhado à esquerda:

Exemplo:

```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <h1 align=left> Título </h1>
</body>
</html>
```

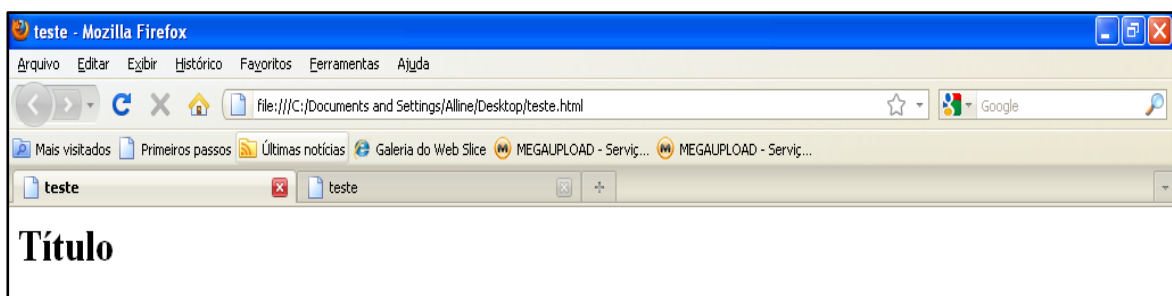
Obs: O alinhamento a esquerda é o padrão para texto.



- **Negrito ():**

Exemplo:

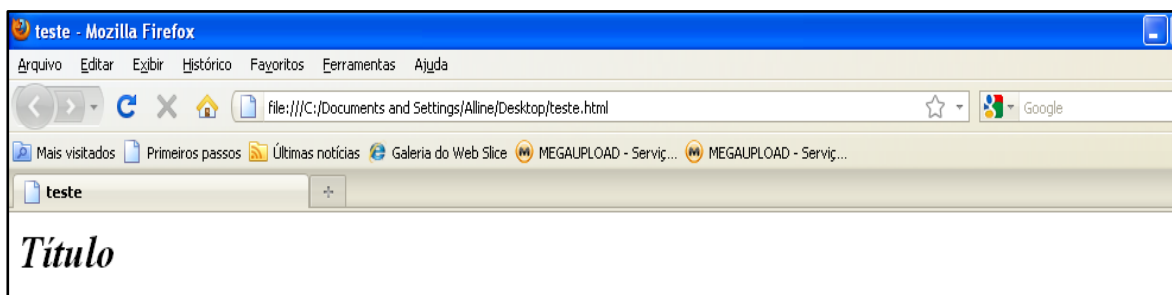
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <h1><b> Título </b></h1>
</body>
</html>
```



- **Itálico (<i>):**

Exemplo:

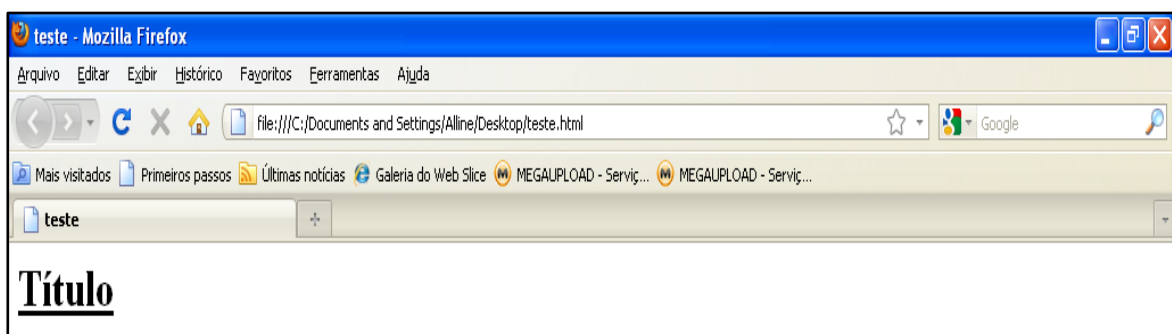
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <h1><i> Título </i></h1>
</body>
</html>
```



- Sublinhado (<u>):

Exemplo:

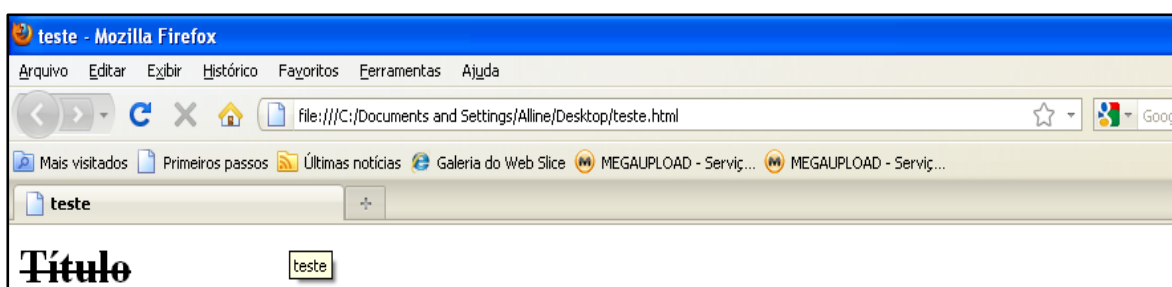
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <h1><u> Título </u></h1>
</body>
</html>
```



- Riscado (<s>):

Exemplo:

```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <h1><s> Título </s></h1>
</body>
</html>
```



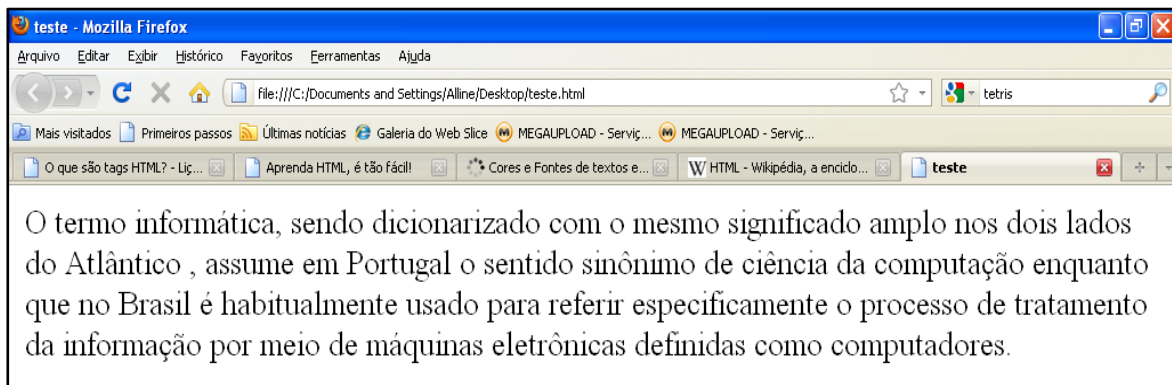
- **Parágrafo (<p>):**

Exemplo:

```
<html>
<head>
  <title> Teste </title>
</head>
<body >
```

<p> O termo informática, sendo dicionarizado com o mesmo significado amplo nos dois lados do Atlântico , assume em Portugal o sentido sinônimo de ciência da computação enquanto que no Brasil é habitualmente usado para referir especificamente o processo de tratamento da informação por meio de máquinas eletrônicas definidas como computadores. </p>

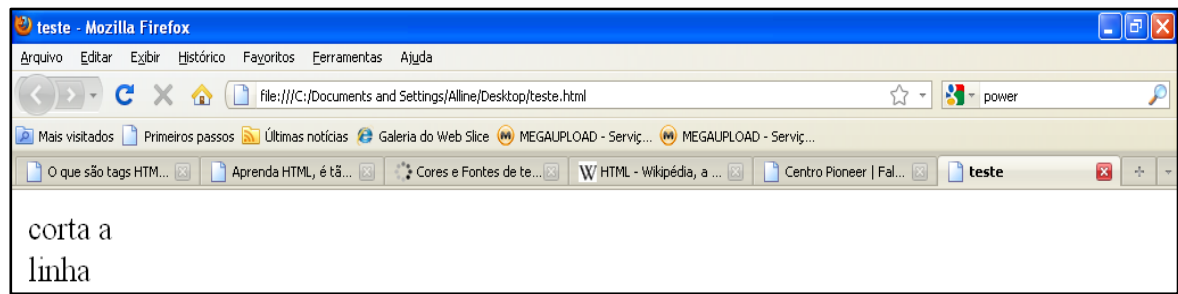
```
</body>
</html>
```



- **Quebra de linha (
):**

Exemplo:

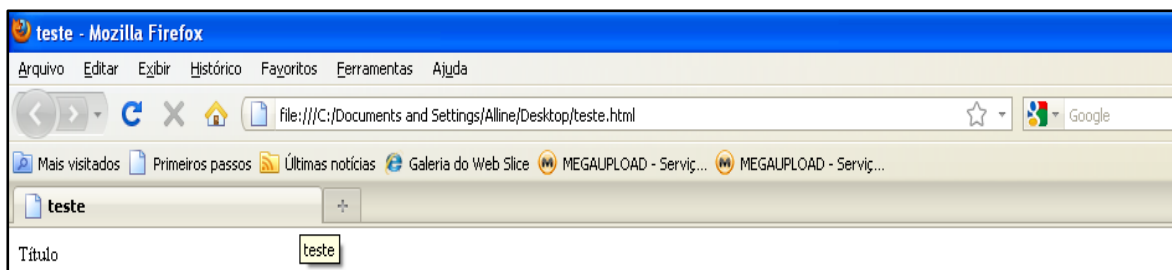
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <p> corta a <br> linha </br></p>
</body>
</html>
```



- Letra pequena (**<small>**):

Exemplo:

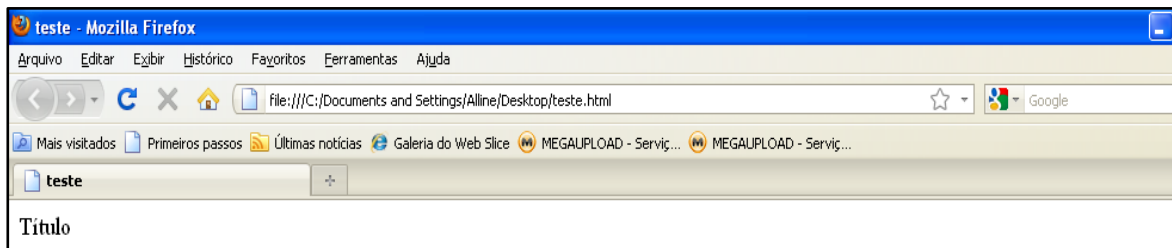
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <small> Título </small>
</body>
</html>
```



- Letra grande (**<big>**):

Exemplo:

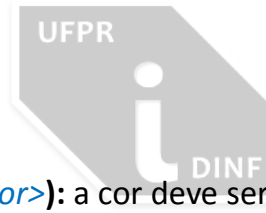
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <big> Título </big>
</body>
</html>
```



- **Hiper-ligação - links - (<a href >):** para um outro local, seja uma página, um e-mail ou outro serviço como download.

Exemplo:

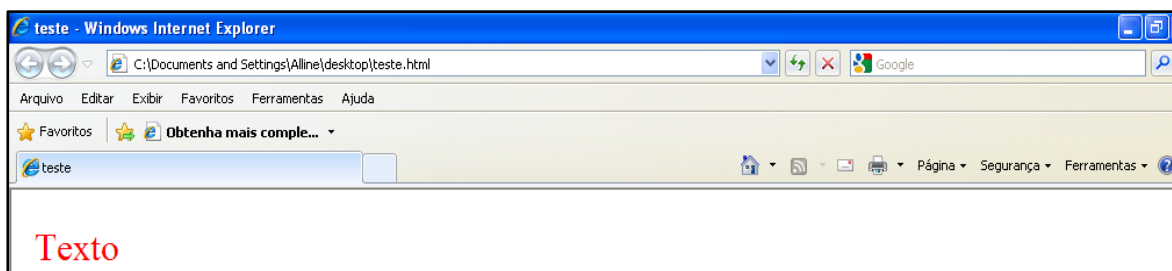
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <a href="http://www.google.com.br/">google</a>
</body>
</html>
```



- **Cor da fonte ():** a cor deve ser escrita em inglês.

Exemplo:

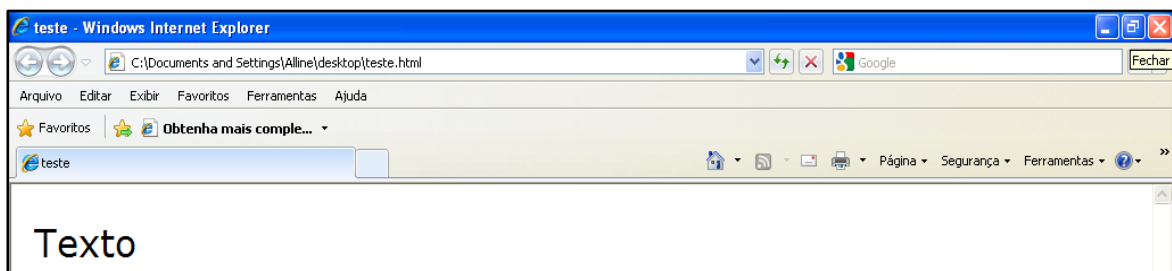
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <font color="red"> Texto </font>
</body>
</html>
```



- **Tipo da fonte ():**

Exemplo:

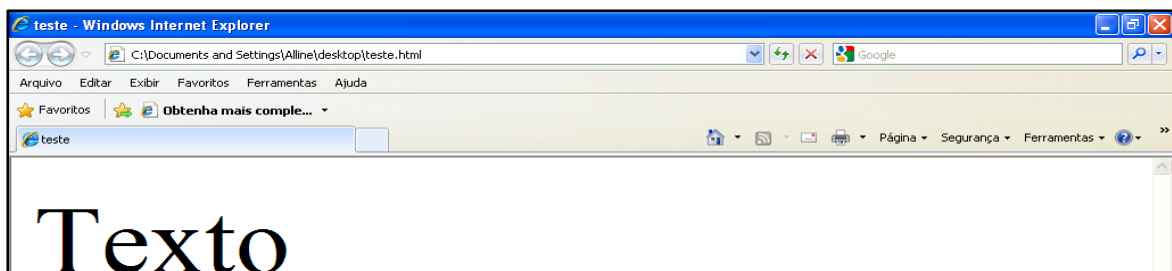
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <font face="verdana"> Texto </font>
</body>
</html>
```



- **Tamanho da fonte ():**

Exemplo:

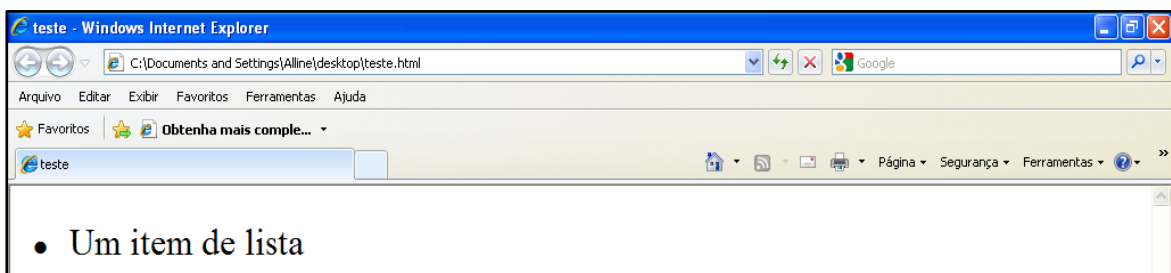
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <font size=12> Texto </font>
</body>
</html>
```



- **Marcadores ():**

Exemplo:

```
<html>
<head>
    <title> Teste </title>
</head>
<body >
    <ul><li> Um item de lista </li></ul>
</body>
</html>
```

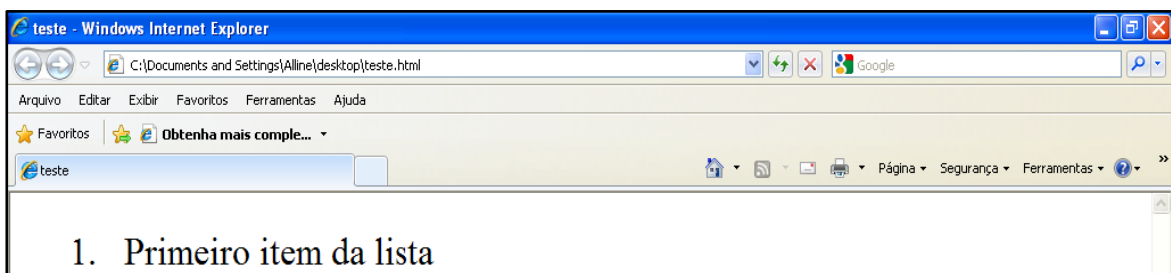


- **Marcadores com parâmetro type (<ol type>):**

- 1 – Numeração iniciada pelo número 1.
- A – Letras do alfabeto em maiúsculas.
- a – Letras do alfabeto em minúsculas.

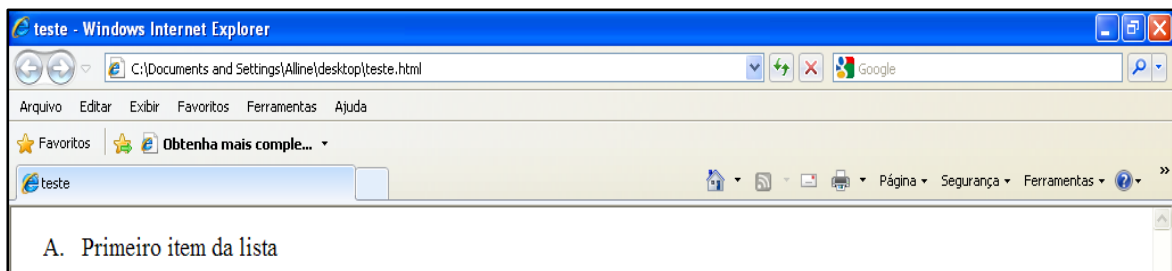
Exemplo 1:

```
<html>
<head>
    <title> Teste </title>
</head>
<body >
    <ol type=1><li> Um item de lista </li></ol>
</body>
</html>
```



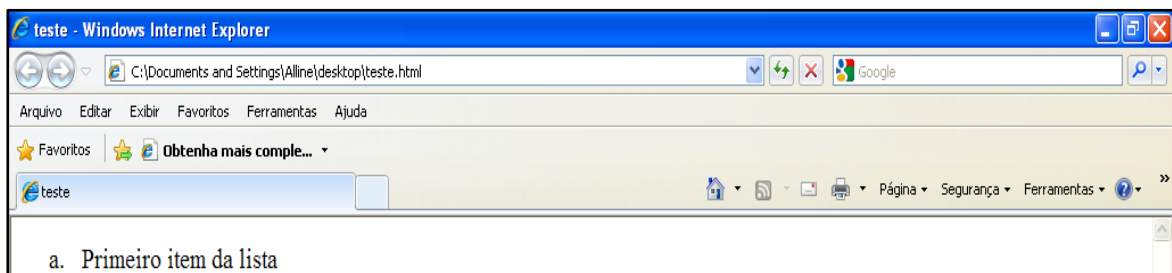
Exemplo 2:

```
<html>
<head>
  <title>Teste</title>
</head>
<body>
  <ol type=A><li>Primeiro item da lista </li></ol>
</body>
</html>
```



Exemplo 3:

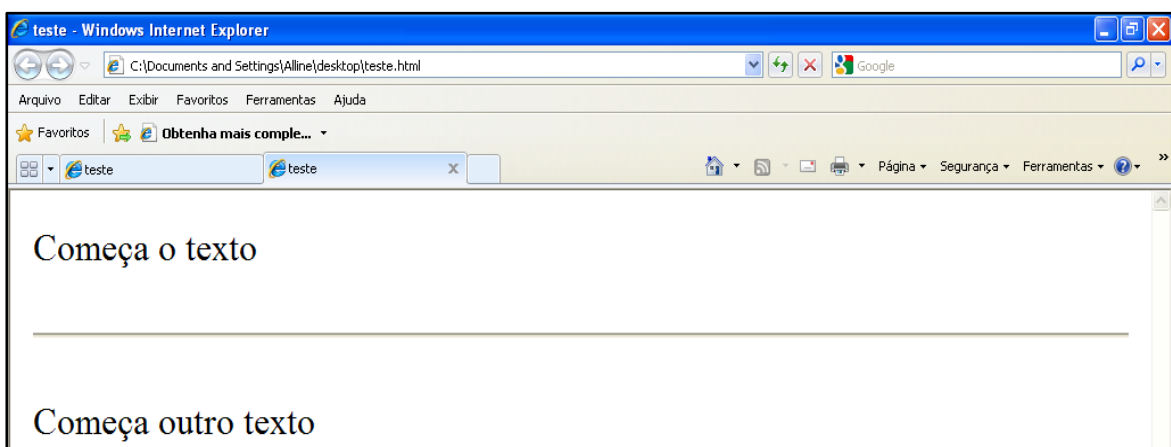
```
<html>
<head>
  <title>Teste</title>
</head>
<body>
  <ol type=a><li>Primeiro item da lista </li></ol>
</body>
</html>
```



- Barra horizontal (`<hr>`):

Exemplo:

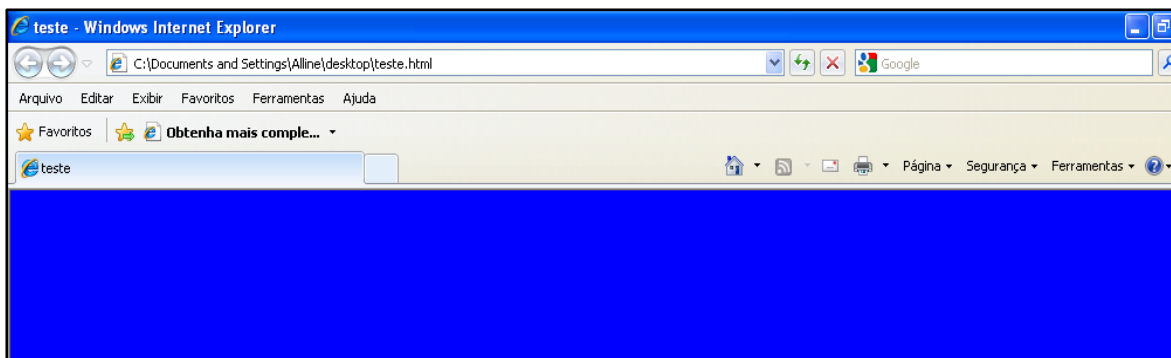
```
<html>
<head>
  <title> Teste </title>
</head>
<body >
  <p> Começa o texto </p> <hr> <p> Começa outro texto </p>
</body>
</html>
```



- Cor de fundo (`<body bgcolor>`):

Exemplo:

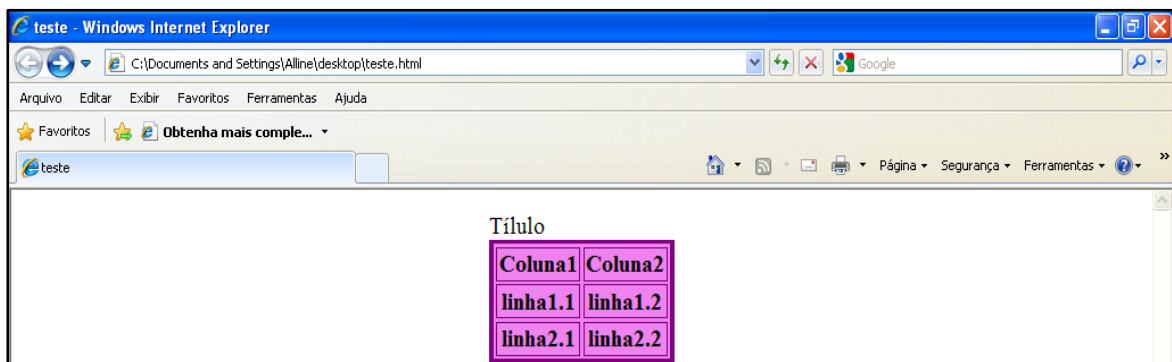
```
<html>
<head>
  <title> Teste </title>
</head>
<body bgcolor="blue">
</body>
</html>
```



- Criação de tabelas (`<table>`):

`<table border>`: tamanho da borda;
`<table bordercolor>`: cor da borda;
`<table bgcolor>`: cor do fundo da tabela,
`<table align=left>`: a tabela fica a esquerda da página;
`<table align=right>`: a tabela fica a direita da página;
`<table align=center>`: a tabela fica no centro da página;
`<table width>`: define a largura da célula da tabela;
`<table height>`: define a altura da célula da tabela;
`<caption>`: título da tabela;
`<table cellpadding>`: insere espaço entre as células;
`<tr>`: delimita uma linha;
`<td>`: delimita uma célula da tabela.

Exemplo:



HTML da tabela do exemplo acima:

```

<html>
<head>
<title>teste </title>
</head>
<body>
  <table border=3 bordercolor=purple bgcolor=violet
  align=center>
    <caption align=center>Título</caption>

    <tr>
      <th>Coluna1</th>
      <th>Coluna2</th>
    </tr>
    <tr>
      <th>linha1.1</th>

```

```

                <th>linha1.2</th>
            </tr>
            <tr>
                <th>linha2.1</th>
                <th>linha2.2</th>
            </tr>
        </table>
    </body>
</html>

```

- **Imagem como plano de fundo (`<body background>`):**

Há duas maneiras de se colocar um imagem como fundo da página:

1. Com a imagem armazenada no seu computador, verificar em qual formato ela está salva. Ex: jpg, gif...

Exemplo:

```

<html>
<head>
    <title>Teste</title>
</head>
    <body background="imagem.jpg">
</body>
</html>

```



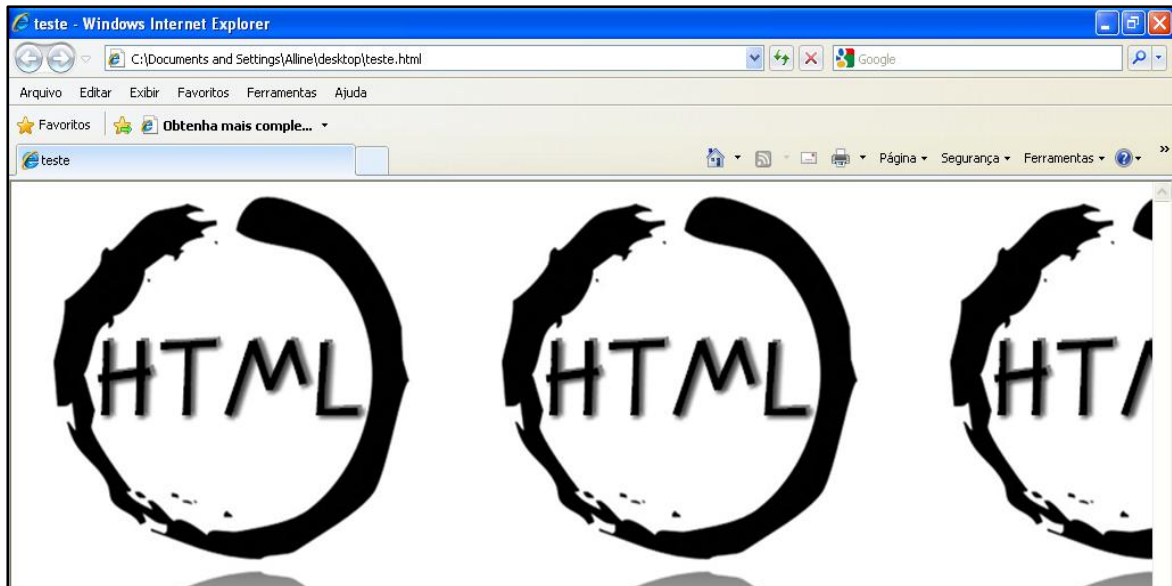
2. Com uma imagem da internet, copiando a url da página:

Exemplo:

```

<html>
<head>
    <title>Teste</title>
</head>
<body
background="http://informatizado.files.wordpress.com/2009/07/html.jpg">
</body>
</html>

```



- **Imagens ():**

Também há dois modos de se colocar uma imagem no seu HTML:

1. Com a imagem armazenada no seu computador, verificar em qual formato ela está salva. Ex: Jpg, gif...

Exemplo:

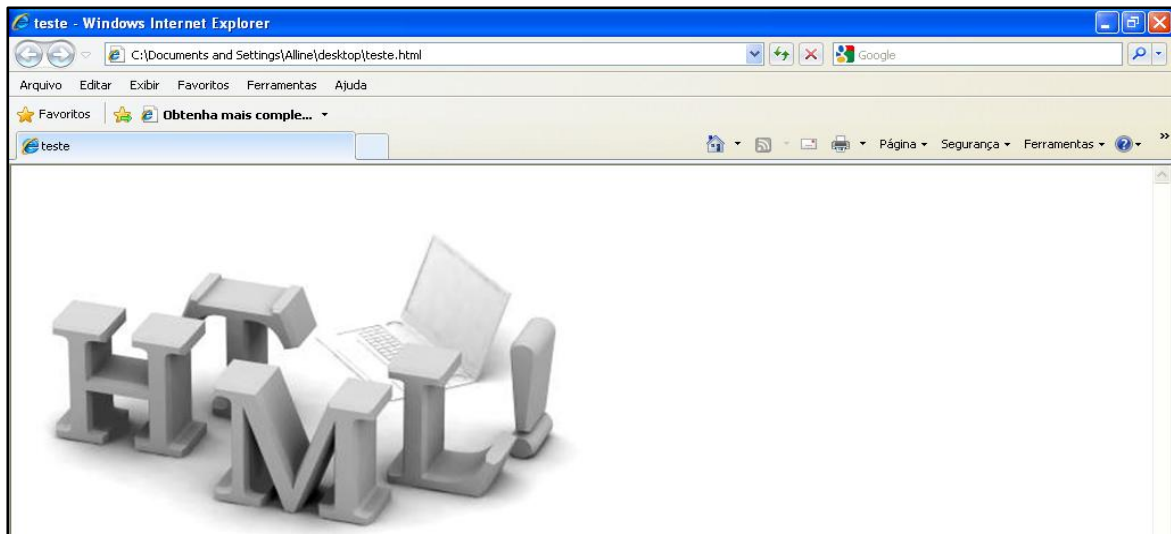
```
<html>
<head>
  <title>Teste</title>
</head>
<body>
  
</body>
</html>
```

2. Com uma imagem da internet, copiando a url da página.

Exemplo:

```
<html>
<head>
  <title>Teste</title>
</head>
<body>
  
```

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



- **Divisões de layout (`<div>`):**

A tag `<div></div>` envolve uma área do texto ou divisão que pode receber parâmetros específicos de alinhamento, como é o caso de `ALIGN`, para alinhar o texto, nesse caso o alinhamento afetará somente o texto dentro da divisão demarcada entre `<div> ... </div>`.

Atualmente ela está sendo muito utilizada por códigos mais avançados. Na verdade o `<div>` não causa nenhuma diferença visual no código, ele é considerado um "container", ou seja, uma espécie de caixa não visual que você pode, através de script, alterar o conteúdo dele, alterando o código HTML dinamicamente, ou então é usado para aplicar um estilo (CSS) em todo o bloco HTML contido dentro do `<div>`. Infelizmente são exemplos complexos demais para um curso de natureza básica como o nosso.

Exemplo:

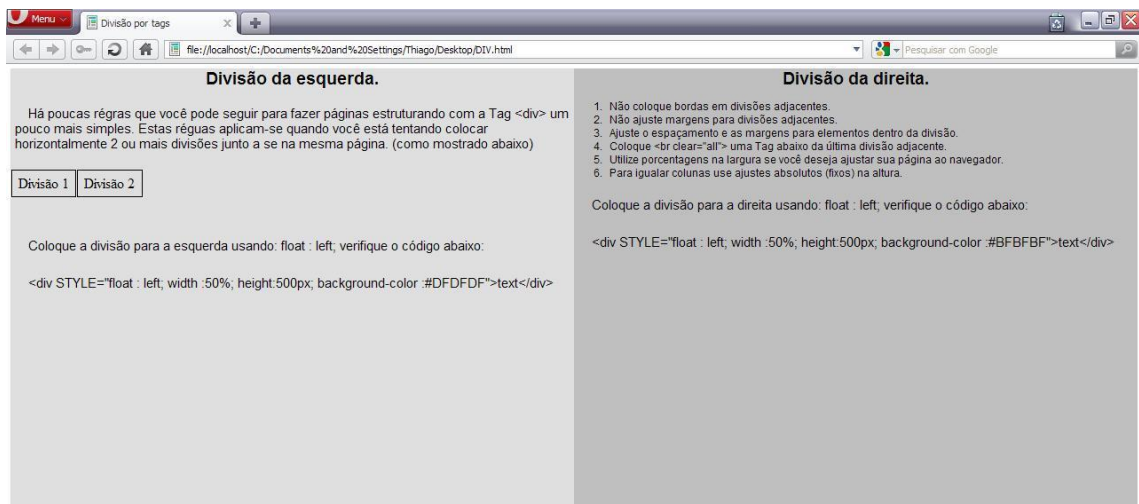
```
<div class="left">
<h1> Divisão da esquerda </h1>
<p> Texto A </p>
</div>
<div class="middle">
<h1> Divisão do meio </h1>
<p> Texto B </p>
</div>
<div class="right">
```

```
<h1> Divisão da direita </h1>
<p> Texto C </p>
</div>
```

Os “class” (Classes) no exemplo acima definem uma forma para o elemento, como cor de fundo, fonte do texto, posicionamento na página, etc...

O nome da classe pode ser qualquer um e é definido pelo desenvolvedor em folhas de estilo, que serão estudadas mais a frente no próximo tema.

Já a tag ``, que eventualmente pode aparecer nos estudos dos alunos, faz o mesmo para uma certa linha. Essa tag é mais utilizada em combinação com folhas de estilo, só para termos uma idéia do que se trata essa tag visualizaremos o exemplo abaixo:



O site acima foi dividido em duas partes, uma cinza escura e a outra um pouco mais clara, cada divisão está dentro de uma tag `<div> ... </div>`.

Você também pode fazer divisões dentro de divisões sucessivamente, onde o posicionamento dos elementos internos tem limite de área delimitada pela divisão que o engloba. Desse modo podemos, por exemplo, fazer uma tabela se utilizando de apenas divisões sucessivas bem posicionadas, é o que os desenvolvedores chamam de “div mania”.

Exemplo:

```
<div id="A">
  <div id="B dentro de A">
    <div id="C dentro de B">
      <div id="D dentro de C">
```



```

...
</div>
<div id="E dentro de C">
...
</div>
</div>
<div id="F dentro de B">
  <div class="G dentro de F">
...
  </div>
  <div class="H dentro de F">
...
  </div>
</div>
</div>
</div>

```

A indentação do código é apenas para deixar claro a idéia de uma divisão dentro de outro, de modo aninhado.

- **Aprendendo mais sobre tabelas (`<table>`)**

Utilizadas como um recurso essencial para o layout geral da página através do posicionamento de imagens e texto, as tabelas são compostas de linhas dentro das quais são colocadas células de conteúdo. O conteúdo de cada célula pode ser texto, imagem ou até mesmo outra tabela.

Uma tabela é delimitada com os marcadores `<table></table>`, sendo que dentro destes marcadores são colocadas as linhas e as colunas da tabela.

Exemplo:

```

<table>
<tr> Indica o início de uma nova linha na tabela
<td> Indica uma coluna na tabela
</td> Indica o final daquela coluna na tabela
</tr> Indica o final da linha na tabela
</table>

```

Existe a possibilidade de se trabalhar com a chamada "CÉLULA TÍTULO" - Uma linha em destaque que pode conter um breve descritivo da tabela. Nesse caso, em vez de `<td>` o marcador de uma "CÉLULA TÍTULO" é indicado por `<th></th>`. Troca-se o D pelo H.

Principais atributos de uma tabela

➤ **Border** (<table **border**="espessura da bora">)

Especifica a presença ou a ausência de borda na tabela bem como sua espessura. No caso de uma tabela sem bordas não é necessário colocar o atributo BORDER.

Exemplo:

```
<table border="10">
```

➤ **Width e Height** (<table **width**="largura da tabela" **height**="altura da tabela">)

Indica o tamanho da tabela. Este caminho pode ser relativo ao tamanho da página, em que a tabela será exibida, ou absoluta. Uma tabela de tamanho relativo é chamada de "TABELA ELÁSTICA" e se estica de acordo com o tamanho da página. Neste caso, a largura e altura da tabela são fornecidas em medida de porcentagem (%).

Obs: 100% indicam que a tabela irá ocupar a tela toda.

Exemplo:

```
<table width="50%" height="100">
```

Repare que a tabela ocupa metade da página na horizontal (WIDHT = 50%), e toda a página na vertical (HEIGHT = 100%).

➤ **Align** (<table **align**="posicionamento">)

Especifica a posição da tabela, que pode ser LEFT, RIGHT ou CENTER.

Exemplos:

```
<table align="left">
```

```
<table align="right">
```

```
<table align="center">
```

➤ **Cellpadding** (`<table cellpadding="valor">`)

Especifica o deslocamento do conteúdo da célula em relação as bordas de cada uma.

Exemplo:

```
<table cellpadding="0">
```

➤ **Cellpadding** (`<table cellpadding="valor">`)

Especifica o espaçamento entre as células da tabela.

Exemplo:

```
<table cellpadding="0">
```

➤ **Bordercolor e bgcolor** (`<table bordercolor = "valor hexadecimal" bgcolor = "valor hexadecimal">`)

Especifica respectivamente a cor da borda e a cor da tabela. As cores devem ser dadas em valores hexadecimais.

Exemplo:

```
<table border="1" bordercolor="#333333" bgcolor="#999999">
```

Obs: As bordas precisam existir. Então é necessário a colocação do atributo BORDER="valor maior que zero", caso contrário, só irá aparecer a cor da tabela (BGCOLOR).

➤ **Background** (`<table background="valor">`)

Especifica a imagem de fundo da tabela.

Exemplo:

```
<table background="casa.jpg">
```

Tendo em vista que se a tabela for maior que a foto de fundo ela se repete.

Principais atributos de uma célula (<td></td>)

Com exceção dos atributos BORDER, CELLPACING e CELLPADDING que são exclusivos da tabela, cada célula possui os mesmos atributos da tabela como: BGCOLOR, BACKGROUND, WIDTH, HEIGHT, e outros.

Além disso, cada célula pode ter seu conteúdo alinhado na horizontal e na vertical.

Exemplo:

```
<td bgcolor="valor"></td>  
<td background="imagem.jpg"></td>  
<td width="largura"></td>  
<td height="altura"></td>
```

Linhas de tabelas (<tr></tr>)

➤ **Alinhamento (<tr align="valor"></tr>)**

Alinhamento horizontal: Os valores podem ser LEFT, RIGHT ou CENTER.

Exemplo:

➤ **Valign (<tr valign="valor"></tr>)**

Alinhamento na vertical: Valores podem ser TOP (topo da célula), MIDDLE (região mediana da célula), BASELINE, (alinha a linha de base do texto da célula com o texto da linha) e BOTTOM (alinha o conteúdo da célula com a região inferior da célula).

Exemplo:

```
<tr valign="baseline"></tr>
```

➤ **Colspan e rowspan** (`<tr colspan="valor" rowspan="valor"></tr>`)

Existem casos de layout da página onde algumas células de uma tabela precisam ser unidas, outras quebradas ao meio, e para isso utiliza-se o atributo SPAN. Sendo que para unir linhas se usa ROWSPAN e para unir colunas utiliza-se COLSPAN.

Exemplo:

```
<tr colspan="3" rowspan="2"></tr>
```



- CSS (Folhas de Estilo) –

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

`h1 {color:blue; font-size: 12px}`

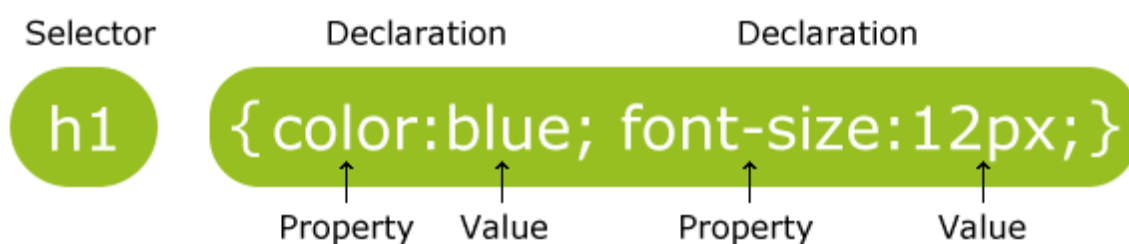


Imagem retirada de "<http://www.w3schools.com/css/selector.gif>"

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, tipografia 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/>

- **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 back-ground 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, paicas, 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 padrão 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>` rotula-dos 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.

- **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).

Exemplo:

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 ou-tro 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 }*.

- **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 frequentemente ignorados, como `</p>`, ``, etc. A falta do `</p>` pode causar o "vazamento" das declarações de estilo para fora do parágrafo em alguns browsers.

- **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 importa-dos). 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 */:

Por exemplo:

```
/* este texto será ignorado, até um asterisco seguido de uma barra seja  
encontrado*/
```

- **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)}
```

- **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.

- **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 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.

- **Como incluo 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.

- **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>*...

- **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}
```

O exemplo 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}
```

- **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.

- **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 padrões 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 padrões do browser: todos os browsers possuem regras comuns para formatar um texto. A especificação HTML não impõe um 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.

- **Styling Background - Definindo propriedades do fundo de elementos HTML**

As propriedades que podem ser modificadas e, ou, acrescentadas são a cor e a imagem através dos comandos background-color e background-image, respectivamente.

Alterando a cor do fundo

Exemplo:

```
body {background-color:#b0c4de;}
```

Que modifica a cor do elemento body para a cor selecionada. As cores podem ser definidas de três maneiras diferentes:

1. pelo nome da cor em inglês(ex: red);
2. por um valor RGB (Red, Green, Blue) (ex: rgb(255,0,0);
3. ou pelo valor hexadecimal de uma cor(ex: #ff0000).

Adicionando imagens

Exemplo:

```
body {background-image:url('paper.gif');}
```

Que adiciona uma imagem ao fundo do elemento body. Para se adicionar uma imagem é preciso passar a URL dela. Nesse exemplo a imagem se encontra no mesmo diretório do arquivo CSS mas caso a imagem esteja em outro lugar, na internet por exemplo, é preciso especificar o caminho completo.

Exemplo:

```
body {background-image:url('http://www.css.com/exemplo/imagem.jpg');}
```

Por padrão, a imagem selecionada é inserida no canto superior esquerdo e é repetida até cobrir totalmente o fundo do elemento. Para definir a posição, a repetição e o nível de fixação da imagem são utilizados os seguintes comandos:

- background-position
- background-repeat
- background-attachment

A posição é dada por duas variáveis (a posição padrão é left top) que podem ser tanto o nome da posição em inglês ou coordenadas baseadas em porcentagens ou píxeis.

A repetição (repeat) ocorre por padrão mas pode ser definida para ocorrer apenas verticalmente, horizontalmente ou simplesmente não ocorrer.

A propriedade attachment define se a imagem roda junto com a página (scroll) ou se ela fica fixa na posição da tela(fixed). Por padrão, as imagens acompanham o rolamento da página.

Background Shorthand



É uma maneira simplificada de se especificar as propriedades vistas em uma única propriedade.

Exemplo:

```
body {background:#ffffff url('img_tree.png') no-repeat right top;}
```

A ordem a ser seguida é: cor, imagem, repetição, fixação e posição sendo que não é necessário que se defina todos os elementos desde que os elementos presentes estejam nessa ordem.

Segue uma tabela com os principais valores das propriedades:

Propriedade	Valores
background	<i>background-color</i> <i>background-image</i> <i>background-repeat background-attachment background-position</i>
background-color	<i>color-rgb</i>

	<i>color-hex</i> <i>color-name</i> transparent
background-image	url(<i>URL</i>) none
background-position	left top left center left bottom right top right center right bottom center top center center center bottom <i>x% y%</i> <i>xpos ypos</i>
background-repeat	repeat repeat-x repeat-y no-repeat
background-attachment	scroll fixed

- **Styling Text - Definindo as propriedades de texto**

As propriedades de texto que podem ser modificadas são: cor, alinhamento, decoração, “indentação” e transformações em letras maiúsculas e minúsculas.

Cores

```
body {color:blue;}
h1 {color:#00ff00;}
h2 {color:rgb(255,0,0);}
```

A cor padrão do texto é definido no campo body mas se outros elementos tiverem suas próprias definições de cores, essas serão prioritárias. No exemplo acima o texto do documento será todo escrito em azul exceto pelos cabeçalhos h1 e h2, que serão verde e vermelho respectivamente.

Alinhamento

```
h1 {text-align:center;}  
p.date {text-align:right;}  
p.main {text-align:justify;}
```

Neste exemplo o cabeçalho h1 será centralizado, os parágrafos da classe *date* serão alinhados à direita e os parágrafos da classe *main* serão justificados.

Decoração

Overline: uma linha sobre o texto;
Line-through: uma linha cortando o texto;
Underline: uma linha sob o texto;
Blink: texto piscante.

```
h1{text-decoration:overline;}  
h2{text-decoration:line-through;}  
h3{text-decoration:underline;}  
h4 {text-decoration:blink;}
```

“Indentação”

Este neologismo que vem do inglês *indentation*, é o nome dado ao espaço entre a primeira palavra de um parágrafo e a margem.

Exemplo:

```
p {text-indent:50px;}
```

Neste caso o parágrafo começará a 50 píxeis de distância da margem.

Transformações

Uppercase: todas as letras são grafadas como minúsculas;
Lowercase: todas as letras são grafadas como maiúsculas;
Capitalize: a primeira letra de cada palavra é grafada como maiúscula.

Exemplos:

```
p.uppercase {text-transform:uppercase;}  
p.lowercase {text-transform:lowercase;}  
p.capitalize {text-transform:capitalize;}
```


Segue uma tabela com outras propriedades de modificação de texto:

Propriedade	Descrição	Valores
direction	Define a direção do texto	ltr (esquerda pra direita) rtl (direita pra esquerda)
line-height	Define a distância entre as linhas	normal <i>número escolhido</i> <i>tamanho em px, cm, etc</i> %
letter-spacing	Aumenta ou diminui a distância entre letras	normal <i>tamanho em px, cm, etc , valores negativos são permitidos</i>
vertical-align	Define o alinhamento vertical de um elemento inserido em uma determinada linha.	baseline (padrão) sub (subscrito) super (sobrescrito) top text-top middle bottom text-bottom <i>deslocamento desejado</i> %
white-space	Define como um espaço em branco de um elemento é tratado.	normal pre (do modo como foi digitado) nowrap (mescla espaços em branco consecutivos em um só e só pula de linha ao encontrar a tag)
word-spacing	Aumenta ou diminui a distância entre palavras	normal <i>tamanho desejado</i>

- **Styling Fonts - Definindo as propriedades da fonte**

No CSS existem dois tipos de famílias de fontes: a genérica e a específica.

A família genérica é composta de diversas fontes que possuem alguma característica em comum, como Serif, Sans Serif e MonoSpace. A família Serif, por exemplo, é composta pelas fontes Times New Roman, Georgia, entre outras.

No caso da família específica, ela é composta por um tipo de fonte apenas, como Times New Roman ou Arial.

Definindo o tipo da fonte

O tipo de fonte a ser utilizada é especificada pela propriedade *font-family*. Podem ser escolhidas diversos tipos de fontes ao mesmo tempo, para o caso do navegador não reconhecer uma delas, separando-as por vírgulas na hora de defini-las. No caso de a família possuir um nome com mais de uma palavra deve-se colocá-la entre aspas.

Exemplo:

```
p{font-family:"Times New Roman", serif;}
```

Neste caso o parágrafo *p* usará a fonte Times New Roman e no caso do navegador não reconhecer esse tipo de fonte será utilizado alguma outra fonte pertencente à família Serif.

Definindo o tamanho da fonte

O tamanho da fonte é definida pela propriedade *font-size*. Para se explicar o uso dessa propriedade é necessário explicar algumas coisas antes.

1. O tamanho padrão da fonte nos navegadores é de 16px (pixels);
2. A unidade mais utilizada para o tamanho é o *em*, sendo 1*em* o tamanho atual da fonte. Para o padrão temos 1*em* = 16px;
3. Combinando esses valores com porcentagens, temos: 1*em* = 16px = 100%;
4. O tamanho da fonte definido para o elemento *body* é o padrão para todo o documento.

Tendo isso em mente, vamos ao exemplo:

```
body {font-size:100%;}  
h1 {font-size:2.5em;}  
h2 {font-size:1.875em;}  
p {font-size:0.875em;}
```

Neste caso a fonte padrão do documento é de 100% = 16px = 1*em*. Em *h1* será utilizada uma fonte de tamanho 2,5 vezes maior do que a do padrão, ou seja, 2.5*em* = 250% = 40px. O mesmo ocorre para *h2* e para *p*.

- **Styling Links - Definindo propriedades de links**

Qualquer uma das propriedades vistas até agora podem ser alteradas nos links: *color, background-color, text-decoration, font-family, font-size, etc.*

O que é exclusivo dos links são mudanças das propriedades de acordo com o estado no qual eles se encontram.

Exemplo:

```
a:link {color:#FF0000;text-decoration:none;}  
a:visited {color:#00FF00;text-decoration:none;}  
a:hover {color:#000000;text-decoration:underline;}  
a:active {color:#0000FF;text-decoration:underline;}
```

Link: refere-se a um link não visitado;

Visited: link visitado;

Hover: mouse sobre o link;

Active: mouse selecionado.

Lembrando que *a* vem da tag html `<a>` com a qual se define links.

Nesse exemplo, teremos:

[a:link](#) será vermelho e sem nenhum traço;

[a:visited](#) será verde e sem nenhum traço;

[a:hover](#) será preto e com underline;

[a:active](#) será azul e com underline.

- **Styling Lists - Definindo propriedades de listas**

A propriedade que pode ser modificada é o tipo de marcador utilizado, podendo ser um tipo pré-definido ou uma imagem qualquer escolhida pelo desenvolvedor (estrelas, corações, smiles, etc).

Lembrando que existem dois tipos de listas em html: listas desordenadas (*ul*) e listas ordenadas (*ol*).

Exemplo:

```
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
ol.c {list-style-type: upper-roman;}
ol.d {list-style-type: lower-alpha;}
ul.e {list-style-image: url('estrela.gif');}
```

Neste caso, listas do tipo:

- a terão como marcador um círculo;
- b terão marcadores do tipo “quadrado”;
- c terão números romanos ordenados (I, II, III...);
- d terão letras minúsculas (a, b, c...);
- e terão como marcador uma imagem definida pelo desenvolvedor.

- **Styling Tables - Definindo propriedades de tabelas**

Todas as propriedades de texto, tais como *color*, *background-color*, *font-size*, *etc*, podem ser alteradas em tabelas. As propriedades específicas para tabela são as referentes ao controle de sua dimensão, aparência e da disposição de seu conteúdo.

Lembrando que em *html* as tabelas são compostas por cabeçalhos (*th*), fileiras (*tr*) e células (*td*). Outro detalhe é que a tabela em si, cada elemento do cabeçalho e as células possuem sua própria borda.

Exemplo de declaração de tabela:

```
table, th, td {border: 1px solid cor;}
th {background-color:red; color:blue; font-size:30px;}
```

Neste caso, a tabela, o cabeçalho e as células terão uma borda com a *cor* desejada e com 1px de espessura. Tanto a espessura quanto a cor são definidas pelo desenvolvedor.

O cabeçalho *th* mostra como é possível alterar outras propriedades dos elementos da tabela.

É possível, se desejado, fazer com que a tabela, o cabeçalho e as células tenham cores e espessuras distintas entre si.

Mesclando bordas

Como cada um dos elementos da tabela possui sua própria borda, o que ocorre é que as tabelas acabam apresentando bordas duplas. Uma maneira de se evitar isso é utilizando a propriedade *border-collapse* que mescla as diferentes bordas em uma só.

Exemplo:

```
table {border-collapse:collapse;}  
table,th, td {border: 1px solid black;}
```

Uma observação a ser feita é que ao utilizar essa propriedade é necessário especificar qual o padrão html que está sendo utilizado.

Exemplo:

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



Altura e largura

A altura e largura de cada um dos elementos da tabela depende do seu conteúdo. Por exemplo, se o maior elemento de uma tabela tiver vinte letras todas as outras células terão uma largura referente a vinte letras. A mesma coisa com relação a altura.

Exemplo:

```
table {width:100%;}  
th {height:50px;}
```

Neste caso, a largura de todos os elementos será de 100%, ou seja, o tamanho do maior elemento mais espaços em branco na mesma quantidade. E o cabeçalho terá uma altura de 50px.

Observação: Se as dimensões especificadas pelo desenvolvedor não forem suficientes para conter o conteúdo (faltar espaço), a tabela irá redefinir sua dimensão conforme o necessário.

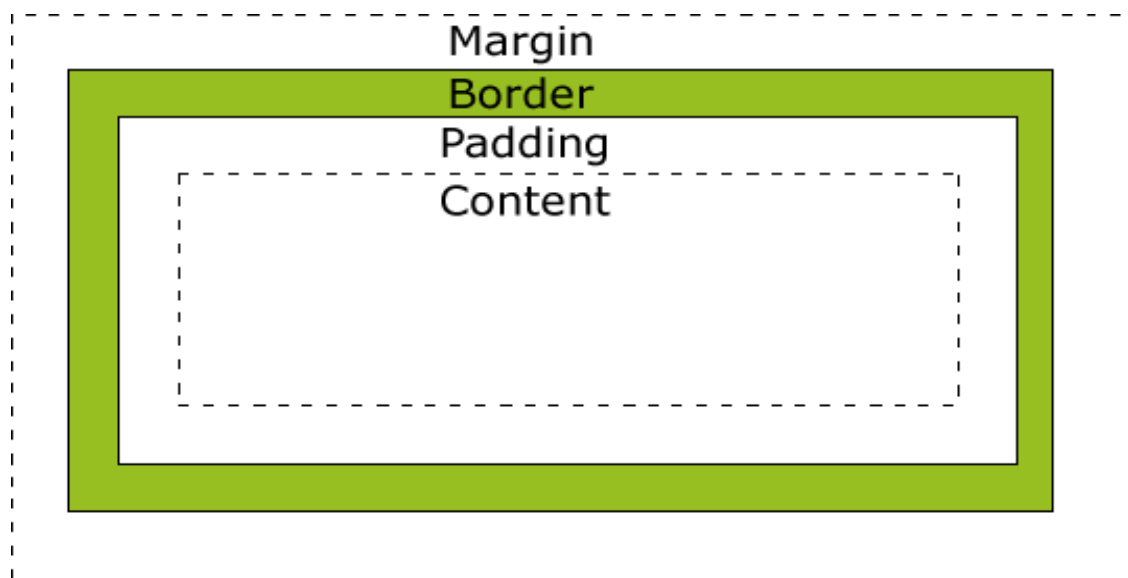
- **CSS Box Model**

Todos os elementos em HTML podem ser considerados “Boxes”, ou caixas. O Termo “Box Model” é usado quando nos referimos ao design e o layout da página.

O Box Model é formado por 4 áreas retangulares que são as seguintes:

- Margens (Margin)
- Bordas (Border)
- Espaçamentos (Padding)
- Conteúdo (Content)

Que fica assim:



(Imagem retirada de http://www.w3schools.com/css/css_boxmodel.asp)

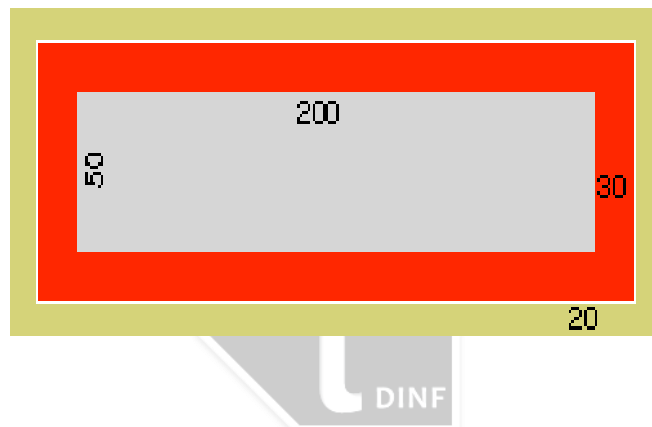
- **Margem (Margin)** - A Margem fica ao redor da borda e não tem uma cor de fundo, é completamente transparente.
- **Borda (Border)** - A borda é afetada pela cor de fundo da box e está ao redor do conteúdo com os espaçamentos.
- **Espaçamentos (Padding)** - Limpa uma área ao redor do conteúdo. E também é afetado pela cor de fundo do Box.

- **Conteúdo (Content)** - O Conteúdo da Box, onde aparecem textos, imagens, etc.

O box model pode ser aplicado em diversas tags do HTML, mas as mais usadas são `<div>`, ``, `<p>`, `<h1>` a `<h6>` e em tags de tabelas como `<table>`, `<th>` e `<td>`.

Tamanho

Para calcular o tamanho do Box temos:



O tamanho do Box (altura e largura) + espaçamento interno + borda + margem.

No exemplo da figura acima temos como largura:

$$20 + 2 + 30 + 200 + 30 + 2 + 20 = 304$$

Para chegar a este cálculo consideramos os 20 px da margem direita e esquerda, os 2 px da borda direita e esquerda, os 30 px do espaçamento direito e esquerdo e mais 200 px da largura do box model.

O calculo da altura é similar: altura + espaçamento interno + borda + margem.

No exemplo da figura acima temos como altura:

$$20 + 2 + 30 + 50 + 30 + 2 + 20 = 154$$

Estilo da Borda:

Alguns tipos:

- **none**: Sem bordas;
- **dotted**: Borda com pontinhos;
- **solid**: Borda solida;
- **double**: Duas bordas;
- **groove**: Borda em 3D, para dentro;
- **ridge**: Borda em 3D, para fora.

Existe a possibilidade de testar essas bordas no site da W3Schools
(http://www.w3schools.com/css/tryit.asp?filename=trycss_border-style)

Largura

A propriedade “border-width” é usado para definir a largura da borda.

A largura é definida em pixels, ou usando conceitos pré estabelecidos: thin, medium, ou thick.

Exemplo:

```
p.one {border-style:solid; border-width:5px;}  
p.two{border-style:solid;border-width:medium;}
```

No site da W3Schools podemos testar o código acima
(http://www.w3schools.com/css/tryit.asp?filename=trycss_border-width)

Cores

A propriedade “Border Color” é usada para definir a cor da borda. A Cor pode ser definida assim:

- **name**: o nome de uma cor, como "red"
- **RGB**: o valor do RGB, como "rgb(255,0,0)"
- **Hex**: um valor hexadecimal, como "#ff0000"

Você também pode definir a cor da borda como “transparent”.

Margem

A margem não tem cor de fundo, e é completamente transparente.

A margem superior, direita, inferior, e esquerda pode ser mudada independentemente usando diferentes propriedades. A propriedade “shorthand” também pode ser usado, para mudar todas as margens de uma vez.

Possíveis Valores:

Valor	Descrição
auto	O Navegador define a margem. O Resultado disso depende do navegador usado.
<i>length</i>	Define uma margem fixa (em pixels, pt, em, etc.)
%	Define a margem em % do elemento

É possível definir margens diferentes nos lados diferentes.

Exemplo:

margin-top:100px;

margin-bottom:100px;

margin-right:50px;

margin-left:50px;

(As margens de cima e de baixo são maiores que as laterais)

Propriedade “Shorthand”

É possível definir todas as margens em uma propriedade apenas, que é chamado de propriedade “Shorthand”.

O mesmo exemplo de cima ficaria assim:

margin:100px 50px;

Podemos ver ambos os exemplos no site w3schools.com:

http://www.w3schools.com/css/tryit.asp?filename=trycss_margin_sides

http://www.w3schools.com/css/tryit.asp?filename=trycss_margin_shorthand

Todas as Propriedades de Margens:

Propriedade	Descrição	Valores
margin	Definir as propriedades da margem com uma declaração	<i>margin-top</i> <i>margin-right</i> <i>margin-bottom</i> <i>margin-left</i>
margin-bottom	Define a margem inferior	auto <i>length</i> %
margin-left	Define a margem esquerda	auto <i>length</i> %
margin-right	Define a margem direita	auto <i>length</i> %
margin-top	Define a margem superior	auto <i>length</i> %

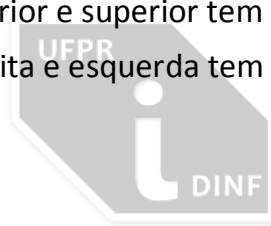
Exemplos:

- **margin:25px 50px 75px 100px;**
 - a margem superior tem 25px
 - a margem direita tem 50px
 - a margem inferior tem 75px
 - a margem esquerda tem 100px

- **margin:25px 50px 75px;**
 - a margem superior tem 25px
 - as margens direita e esquerda tem 50px
 - a margem inferior tem 75px

- **margin:25px 50px;**
 - as margens inferior e superior tem 25px
 - as margens direita e esquerda tem 50px

- **margin:25px;**
 - todas as margens tem 25px



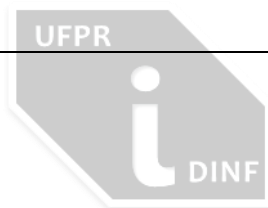
Espaçamentos

O Espaçamento (Padding) , segue as mesmas regras da margem. Por exemplo, para definir um espaçamento com todos os lados com 25px, fazemos assim:

padding:25px;

Todos os outros exemplos de margens são validos para padding:

Propriedade	Descrição	Valores
padding	Definir as propriedades de espaçamento com uma declaração	<i>padding-top</i> <i>padding-right</i> <i>padding-bottom</i> <i>padding-left</i>
padding-bottom	Define o espaçamento inferior	<i>length</i> <i>%</i>
padding-left	Define o espaçamento esquerdo	<i>length</i> <i>%</i>
padding-right	Define o espaçamento direito	<i>length</i> <i>%</i>
padding-top	Define o espaçamento superior	<i>length</i> <i>%</i>



- JavaScript -

O JavaScript foi desenvolvido inicialmente pela Netscape e então se intitulava LiveScript. Adotado no fim do ano de 1995, pela firma Sun (que também desenvolveu o Java), ele se tornou o JavaScript.

Trata-se uma linguagem de script que incorporada nos tags HTML, permite incrementar a apresentação e interatividade das páginas WEB.

Estes scripts vão ser gerados e executados pelo próprio browser (Firefox, Netscape, Internet Explorer) e estas instruções serão executadas diretamente e, sobretudo, sem atrasos.

Obs.: Não confunda, JavaScript **NÃO** é Java.

O JavaScript é um código integrado diretamente na página HTML, interpretado pelo browser apenas no momento da execução e o código é visível, portanto, a segurança é muito baixa. Já o Java, é um módulo distinto da página HTML, onde o código é compilado antes de sua execução e como é um código compilado e não-visível aos outros, é o tipo de programação segura.

Javascript é mais simples realizar visto que o código que se insere diretamente nas páginas HTML como, por exemplo, um simples editor de texto como o Notepad. O Java necessita de uma compilação prévia do código. Porém, o campo de aplicação do Javascript é um tanto limitado, enquanto que com o Java pode-se fazer, em princípio, tudo.

- **Incorporando o JavaScript em HTML**

O código em JavaScript do lado cliente é incorporado em documentos HTML de diversas maneiras:

- Entre um par de marcas `<script>` e `</script>`;
- Em um tratador de evento, especificado como valor de um atributo de HTML como onclick ou onmouseover;
- Com o corpo de um URL que usa o protocolo especial JavaScript.

O tag <script>

Como já foi visto, o Javascript insere-se numa página WEB.

A linguagem HTML utiliza os tags para “dizer” ao browser para inserir uma porção de texto em negrito, em itálico, etc.

Na lógica da linguagem HTML, é então necessário avisar ao browser através de um tag que o texto que segue é script e, mais importante que isso, que esse script é JavaScript. Usa-se o tag:

<script language =“javascript”>

Também é necessário informar ao browser do fim do script. Usa-se o tag:

</script>

Comentários

Será útil incluir comentários pessoais no código Javascript.

É mesmo recomendado como para todas as linguagens de programação. O Javascript utiliza as convenções utilizadas em C e C++, ou seja:

// comentário.

(Tudo o que está escrito entre o // e o fim da linha será ignorado.)

Também é possível incluir comentários em diversas linhas com o código:

/ comentário em diversas linhas */*

Obs.: JavaScript é “*case sensitive*”. Assim, será necessário escrever “*alert()*” e não “*Alert()*”. Para a escrita das instruções JavaScript, utilizaremos o alfabeto ASCII clássico (a 128 caracteres) como em HTML. As aspas “” e o apóstrofo “’” fazem parte da linguagem JavaScript. Pode-se utilizar uma ou outra forma na condição de não às misturar. Assim, alert(“...”) dará uma mensagem de erro.

- **Orientação a objetos**


Em JavaScript temos uma lógica muito simples, bastando sempre seguir o caminho dos **objetos**. Isso torna a programação mais simples e objetiva, ou seja, é o que chamamos de programação orientada a objetos. Isso quer dizer que em JavaScript, todos os elementos de uma página são tratados como objetos. E para que a programação funcione, esses objetos são sempre agrupados de acordo com o seu tipo e/ou finalidade.

- **Manipulação de objetos**

A linguagem JavaScript é baseada em objetos. Portanto, se vamos programar em JavaScript, vamos manipular objetos.

Como existe uma hierarquia a ser seguida e cada objeto tem seu lugar, ao manipularmos objetos estamos na verdade fazendo com que cada objeto se torne propriedade de outro.

Vamos ver um exemplo dessa hierarquia:

- 
- 1° – BROWSER (navegador)
 - 2° – MATH / DATE / WINDOW / NAVIGATOR / STRING
 - 3° – DOCUMENT
 - 4° – LINK / FORM / ANCHOR
 - 5° – SELECT / BUTTON SUBMIT / TEXTAREA / RADIO / CHECKBOX

- **Propriedades dos objetos**

Cada objeto manipulado em JavaScript possui uma propriedade ou característica.

Para entendermos melhor isso, vamos ver um exemplo:

No caso do objeto **form** (formulário), vemos uma propriedade, uma característica do objeto **document** (documento) como visto na lista do artigo anterior. (document = nível 3 / form = nível 4).

Então, para entender bem essa parte, sabemos que um objeto de menor hierarquia que o outro é uma propriedade.

Portanto, na hora de programar vamos ver:

nomedoobjeto.propriedade

Métodos de objetos

Além das propriedades (características), os objetos podem conter métodos.

Os métodos são funções pré-definidas pela linguagem JavaScript que irão executar uma operação.

Por exemplo:

Vamos supor que você faz uma programação simples que consiste em ter um documento na tela em que o usuário escreve um texto e ao clicar em um botão esse texto é exibido em outro lugar. O ato de escrever, clicar e o aparecimento desse texto pode ser chamado de **método**. E um método estará sempre associado a um objeto no documento.

Em quase todos os casos na programação com JavaScript os métodos são usados para alterar o valor de uma propriedade ou executar tarefas específicas. Vamos ver a sintaxe de utilização de um método:

nomedoobjeto.método(argumento), onde:

- *nomedoobjeto* = objeto a ser utilizado e que sofrerá uma ação do método.
- *método* = nome de indentificação.
- (*argumento*) = expressão ou valor opcional que será usado para alterar o objeto.

Eventos

Em **JavaScript**, assim como em qualquer linguagem orientada a objetos, é comum a manipulação de eventos que nada mais é que uma ação que executa um determinado procedimento. Por exemplo, consideramos um evento o ato de o usuário clicar em um botão. Portanto, entenda um evento como qualquer ação iniciada pelo usuário.

A utilização dos eventos se dá dentro das próprias tags HTML e sua sintaxe tem a seguinte formação:

```
<TAG HTML nomedoevento="instrução JavaScript">
```

Se for necessário utilizar mais de um **comando JavaScript** na mesma tag, utilizamos então o ponto e vírgula (;) para separá-los.

Por exemplo:

```
<TAG HTML nomedoevento="instrução JavaScript1;instrução  
JavaScript2">
```

Alguns exemplos de eventos são:

- blur ou onBlur = quando o usuário muda o foco de um objeto.
- change ou onChange = quando o usuário muda o valor de um objeto.
- click ou onClick = quando o usuário clica sobre um objeto.

Para entender melhor vamos ver um exemplo simples de evento que é uma mensagem assim que o usuário entra na página:

```
<html>  
<head>  
<title> Eventos </title>  
</head>  
<body onLoad=javascript:alert('Welcome!')>
```

Copie esse código e cole em um editor de texto, depois salve com o nome que quiser, porém, lembre-se que a extensão tem que ser *“.html”*. Por exemplo: *pagina.html*. Em seguida abra o documento no seu navegador.

No exemplo acima, vimos o evento **onLoad** que quer dizer “quando a página é carregada”. Junto da instrução **alert()** que tem como função exibir uma caixa de diálogo do Windows com a mensagem que definimos entre aspas, permitindo ao usuário fechá-la ao clicar em **OK**.

Nome de variáveis

O nome de uma variável pode tanto iniciar-se por uma letra como por caracteres especiais e até números. Além da diferenciação de letra maiúscula e minúscula. O importante é saber que se uma variável for definida sem a

instrução **var** ela se torna global, ou seja, valem para todas as funções que estejam dentro do script da mesma página.

Quando utilizamos a instrução **var**, a variável então passa a ser local, ou seja, utilizada apenas para aquela função onde foi declarada.

Agora não vamos mais nos prender a parte teórica, mesmo porque, já falamos sobre o básico da teoria aqui, você pode encontrar muito mais material teórico pela rede, uma vez entendida essa parte básica. Gostaria agora de falar da parte prática:

Vamos entender como funciona um código de nível iniciante, mas que já mostra o que pode ser feito e **como funciona a linguagem JavaScript**.

Primeiro vamos ao código:

```
<html>
<head>
<title> JavaScript </title>
<script type="text/javascript">
function calcula(){
var primeiroNumero = parseFloat(document.form1.texto1.value);
var segundoNumero = parseFloat(document.form1.texto2.value);
alert(primeiroNumero + segundoNumero);
}
</script>
</head>
<body>
<form name="form1">
Primeiro Numero: <input name="texto1" size="3"><br>
Segundo Numero: <input name="texto2" size="3"><br>
Clique aqui para o resultado: <input type="button" value="Calcule"
onclick="calcula()">
</form>
</body>
</html>
```

Copie esse código e cole em um editor de texto, depois salve com o nome que quiser, porém, lembre-se que a extensão tem que ser **".html"**.
Por exemplo: *pagina.html*

Em seguida abra o documento no seu navegador.

Agora vamos dividir o código e procurar entender cada parte:

<script> = Aqui começa de fato o JavaScript.

function calcula(){ = Abertura da função que faz o cálculo.

var primeiroNumero = parseFloat(document.form1.texto1.value); = variável1.

var segundoNumero = parseFloat(document.form1.texto2.value); = variável2.

alert(primeiroNumero + segundoNumero); = Abre a janela de alerta do Windows com o resultado da soma entre o primeiro número e o segundo número representados pelas variáveis 1 e 2.

} = Fecha a função.

</script> = Fecha o JavaScript.

<form> = Abre o código do formulário. (o formulário é responsável por fazer os campos de texto e botão funcionarem como uma coisa só).

**Primeiro Numero: <input type="text" size="3">
** = Primeiro campo de texto.

**Segundo Numero: <input type="text" size="3">
** = Segundo campo de texto.

Clique aqui para o resultado: <input type="button" value = "Calcule" onclick="calcula()"> = Responsável por chamar a função do JavaScript que faz o cálculo e abre a janela do Windows com a resposta.

</form> = Fecha o formulário

Fontes e referências:

❖ Funcionamento de aplicações Web:

<http://pt.wikipedia.org/wiki/navegador>
<http://www.pt-br.html.net/tutorials/html>
<http://hospedagemdesites.com.br>

❖ HTML e CSS:

[http:// www.w3schools.com/](http://www.w3schools.com/)

❖ JavaScript

Apostila “JavaScript – Aplicações Interativas para a Web” de Adriano Gomes Lima - <http://www.gico.com.br/site/pdf/JavaScript.pdf>;

Apostila “JavaScript Para Iniciantes” -
<http://www.scriptbrasil.com.br/download/apostila/99/>;

Tutorial “Javascript Básico Para Iniciantes”
<http://www.ikaro.net/br/2009/08/objetos-javascript.html>.

