
PROGRAMAÇÃO DE SCRIPTS HTML

RESUMO DAS AULAS

RESUMO DAS AULAS DE HTML

Hiper Text Markup Language.....	3
PARTE 1 - Introdução	3
1 - Conceitos Básicos.....	3
2 - Estrutura de uma Página Web HTML.....	4
3 - Primeiros Elementos	4
4 - Parágrafos.....	5
5 - Configurando Textos.....	6
PARTE 2 - Links e Imagens	7
6 - Documentos com <i>Hiperlinks</i>	7
7 - Exibição de Imagens	7
PARTE 3 - Configurando Listas	8
8 - Listas.....	8
PARTE 4 - Melhorando a Aparência	9
9 - Cores.....	9
10 - Formatando Parágrafos	9
PARTE 5 - Âncoras	10
11 - Utilizando Âncoras	10
PARTE 6 - Tabelas	10
12 - Utilizando Tabelas.....	10
PARTE 07 - Frames	12
13 - Utilizando Frames	12
PARTE 08 - Formulários	13
14 - Utilizando Formulários.....	13
PARTE 09 - Metadados.....	17
15 - Definindo Metadados.....	17
PARTE 10 - Cascading Style Sheets (CSS).....	18
16 - Introdução ao CSS	18
17 - Aplicando Estilos	19
PARTE 11 - Design Web Responsivo	21
16 - Introdução à Responsividade	21

Hiper Text Markup Language

Este material é apenas um resumo para acompanhamento dinâmico das aulas ministradas, sendo necessário o acompanhamento das aulas e as orientações do professor.

PARTE 1 - Introdução

1 - Conceitos Básicos

a. **Página web** é um documento composto basicamente de textos e códigos especiais chamados **tags** (etiquetas), que possibilitam a exibição do documento em um **navegador** (*browser*) na internet. Pode conter imagens, sons, animações, vídeos, links e formulários. Basicamente são escritos em linguagem de marcação de hipertextos **HTML**, apesar de receberem implementos em linguagens de *scripts* - como **Javascript** por exemplo - linguagens dinâmica de programação - como **PHP** por exemplo - dentre outras.

b. **World Wide Web** é a rede mundial de computadores, conhecida como **Web**, **www** ou **W₃**, é um conjunto de documentos multimídia conectados através de **hiperlinks** que permitem o deslocamento entre documentos na internet. O ato de seguir hiperligações é chamado "navegar" ou "surfar" na Web.

c. **URL (Uniform Resource Locator - Localizador Uniforme de Recursos)** é o endereço na Web no qual se encontram recursos como uma página web, um arquivo de computador ou um dispositivo periférico (impressora, equipamento multifuncional, unidade de rede etc.). Essa rede pode ser a internet, uma rede corporativa (intranet). Um URL completo possui a seguinte estrutura:

```
protocolo://domínio:porta/caminho/recurso  
?parâmetro=argumento#fragmento
```

- **protocolo** pode ser HTTP, HTTPS, FTP, etc.
- **domínio** é o endereço do servidor que hospeda o documento/recurso solicitado, sendo **:porta** opcional como um ponto lógico no qual se executa a conexão com o servidor.
- O **caminho** especifica o local (pasta, sub-pastas, diretórios) onde se encontra o **recurso**, dentro do servidor.
- **parâmetro=argumento** é um conjunto de pares de parâmetros com seus respectivos argumentos (senha=123, por exemplo). É opcional, sendo uma string enviada ao servidor para que seja possível filtrar o recurso.
- O **fragmento** é uma parte opcional, uma âncora, ou posição específica dentro do recurso.

d. **Endereço IP (Internet Protocol ou Protocolo de Internet)**, é uma identificação de um dispositivo (computador, impressora, etc.) em uma rede, sendo este o meio em que as máquinas usam para se comunicar na Internet. Cada conexão em uma rede possui um IP único. Para facilitar a navegação, comumente utilizamos endereços de domínio, tal como **www.google.com**, sendo este convertido em um endereço IP pelo **DNS (Domain Name System)**. Os endereços IP são dividido

em estruturas de tamanho fixo chamadas de **classes de endereço**. As principais são as classes A, B e C:

Classe	Gama de Endereços	Qtde. de Endereços
A	1.0.0.0 até 127.0.0.0	16 777 216
B	128.0.0.0 até 191.255.0.0	65 536
C	192.0.0.0 até 223.255.255.0	256

- Endereços da **Classe A** permitem menos **redes** e mais **hosts** por rede, enquanto que os endereços da **Classe C** permitem mais **redes** e menos endereços (**hosts**) disponíveis em cada rede.

e. **DNS (Domain Name System - Sistemas de Nomes de Domínio)** é um sistema de gerenciamento de nomes para computadores, serviços ou qualquer recurso conectado à internet. Comumente, traduz os nomes de domínios - que são facilmente memorizáveis pelos usuários - para os endereços IP numéricos. Este processo chama-se **resolução de nome**. O banco de dados onde se armazenam **DNS** é distribuído, assim seu tamanho é ilimitado e o desempenho não se degrada muito quando se adicionam mais servidores.

f. **TCP/IP** é um conjunto de protocolos de comunicação entre computadores em rede, sendo **TCP (Transmission Control Protocol - Protocolo de Controle de Transmissão)** e o **IP (Internet Protocol - Protocolo de Internet)**. O conjunto de protocolos pode ser visto como um modelo de camadas onde cada camada é responsável por um grupo de tarefas, fornecendo um conjunto de serviços bem definidos para o protocolo da camada superior. As camadas mais altas, estão logicamente mais perto do usuário e lidam com dados mais abstratos, sendo os protocolos de camadas mais baixas para tarefas de menor nível de abstração. A **Tabela 1** mostra as camadas e suas aplicações.

Camadas	Aplicações
1ª e 2ª Interface com Rede	Camadas físicas pois tratam-se das tecnologias usadas para as conexões. Exemplos: Wi-Fi, Modem.
3ª Internet	Camada responsável pela interconexão entre as redes locais. Exemplo IP.
4ª Transporte	Camada que controla a comunicação ponto-a-ponto, host-to-host. Exemplo TCP.
5ª, 6ª e 7ª Aplicação	Camadas que contém os protocolos para um serviço específico de comunicação de dados no nível processo-a-processo, como um navegador deve se comunicar com um servidor da web. Exemplos: HTTP, HTTPS, FTP, DNS.

Tabela 1

g. **HTTP (Hypertext Transfer Protocol - Protocolo de Transferência de Hipertexto)** é um protocolo de comunicação de requisição-resposta no modelo computacional cliente-servidor (em uma das camadas de aplicação), utilizado para sistemas de informação de hipermídia. Ele é a base para a comunicação de dados da **World Wide Web**. **Hipertexto** é o texto estruturado que utiliza ligações lógicas (hiperlinks) entre pontos que contenham texto. Um navegador pode ser o cliente e uma aplicação em um computador que hospeda um **site** da web pode ser o servidor. O cliente submete uma mensagem de requisição **HTTP** para o servidor. O servidor retorna uma mensagem resposta para o cliente.

1.1 - Visualizando Páginas Web

Para visualizar uma página web é preciso digitar uma URL em um navegador instalado na máquina onde se deseja exibir a página web. Primeiramente, a parte da URL referente ao servidor de rede (**domínio**) é separada e transformada em um endereço IP, pelo processo de resolução de nome (DNS). Assim, o navegador estabelece uma conexão TCP-IP com o servidor web localizado no endereço IP retornado.

O próximo passo é o navegador enviar uma requisição HTTP ao servidor para obter o recurso indicado pela parte restante da URL. Em seguida, o texto HTML é recebido e **interpretado** pelo navegador. **Não é compilado**. A partir daí, requisições adicionais são feitas para carregar figuras, arquivos de formatação (como .CSS), arquivos de *script* (como .JS) entre outros recursos que compõem a página. O navegador, então, renderiza (constrói) a página na tela.

1.2 - Utilizando um Navegador

Não é preciso estar conectado à internet para se utilizar um navegador, portanto, é muito simples testar as páginas web que estiverem sendo editadas, bastando para isso executar o navegador em um micro e carregar o programa editado.

Às vezes, acontece de uma tag ser reconhecida por um determinado navegador e não ser por outro. Para que isso seja testado, é interessante que se tenha instalado em seu micro pelo menos os dois principais Navegadores do mercado, e testar sua Página Web nos dois. Em caso de incompatibilidade, o melhor é escrever os comandos duas vezes (uma vez com a sintaxe de um navegador e a outra vez com a sintaxe do outro navegador).

2 - Estrutura de uma Página Web HTML

Para escrever uma página web em HTML, o programador deve digitar o código-fonte em um editor de textos qualquer, desde que permita salvar com a extensão .HTM ou .HTML, seguindo as regras de sintaxe do HTML.

O código-fonte de uma página HTML pode ser digitado com caracteres maiúsculos ou minúsculos, pois o HTML não é *key sensitive*. As *tags* podem ser digitadas na mesma linha ou em linhas subsequentes. O programador deve sempre estar atento com abertura e fechamento de *tags* e aspas. O HTML possui estrutura linear e não compilada, portanto não haverá lista de erros.

2.1 - Sintaxe das Tags

Podem ser especificadas em pares ou individuais e estão sempre envolvidas pelos sinais < > para abertura da tag e < / > para fechamento, também chamados de containeres. <TAG> texto </TAG>

Exemplo:

Texto em negrito .

Nem todas as tags precisam de fechamento, como por exemplo, a tag para quebra de linha:

Texto em uma linha.

Texto na linha seguinte.

Neste resumo de aulas, o estudo do HTML considera a utilização da versão **HTML 4.01**, mas inclui também a nova versão **HTML 5**. A maioria das tags do HTML 4.01 se aplicam ao HTML 5, porém é necessária a utilização de estilos em CSS para configurar seus parâmetros. Para

tags ou parâmetros que não se aplicam no HTML 5 o símbolo ☒ é indicado.

2.2 - Estrutura Básica da Página Web

Nem todas as *tags* são obrigatórias, mas basicamente o código fonte HTML segue a seguinte estrutura:

```
<!DOCTYPE html>
<HTML>
<HEAD>
    <!-- comandos de cabeçalho e estilo -->
    <TITLE>Título da Página</TITLE>
</HEAD>
<BODY>
    <!-- montagem do corpo da página -->
</BODY>
</HTML>
```

3 - Primeiros Elementos

a. <!DOCTYPE html>

Todo documento HTML deve conter um DOCTYPE em seu início, antes da tag <HTML>, para informar ao navegador qual é a versão do HTML que está sendo utilizado. Se não for declarado, o navegador tentará adivinhar qual é o DOCTYPE, podendo provocar uma renderização defeituosa. No HTML5 o DOCTYPE é referenciado como um cabeçalho inútil mas necessário, apenas para ativar o modo padrão de renderização em navegadores comuns, isso porque o HTML5 não é baseado em SGML, usando o DOCTYPE apenas para seleção de modo sem utilizar DTD (Declaração do Tipo de Documento). A sintaxe é apenas:

```
<!DOCTYPE html>
```

Até o HTML 4.01, o DOCTYPE necessitava-se fazer referência a um DTD, podendo ser DOCTYPE **Strict**, **Transitional** ou **Frameset**, como segue:

- **HTML 4.01 Strict:** código precisa ser escrito de forma perfeita, sem erros ou esquecimentos. Não utiliza tags e atributos de apresentação como , <BODY BGCOLOR> por exemplo, mas sim utilizar CSS para definir estilos de apresentação.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

- **HTML 4.01 Transitional:** para ajudar os web designers na aprendizagem, menos sensível a alguns erros e esquecimentos ao usar *tags* e atributos de apresentação. É como uma transição entre a forma antiga de codificar para a forma nova e correta.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
```

- **HTML 4.01 Frameset:** seguindo a mesma ideia do transitional, porém para a construção de páginas utilizando utilização de FRAMES.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Frameset//EN" "http://www.w3.org/TR/html4/
frameset.dtd">
```

b. <HTML> ... </HTML>

Especifica que dentro dessas tags há comandos HTML. Representa a raiz de um documento HTML. Todos os outros elementos são descendentes de <HTML>.

c. <HEAD> ... </HEAD>

Área destinada às tags relacionadas com a identificação da página e formatação de estilos. Sua utilização não é obrigatória, entretanto, há tags que necessitam ser declaradas nesta área, como o <TITLE> por exemplo. Também no <HEAD> são incluídos metadados, <META>, que são dados de alto nível que não são exibidos na tela, mas exercem influência na execução da página.

```
<HEAD>
    <META charset="UTF-8">
</HEAD>
```

O exemplo acima, mostra a definição do conjunto de caracteres UTF-8 que permite exibir caracteres da língua portuguesa.

d. <TITLE> ... </TITLE>

Especifica o título da página que aparecerá na aba do navegador. A digitação incorreta desta tag acarretará problemas na exibição da página.

e. <BODY> ... </BODY>

Especifica o início e o fim do corpo principal da página web. Há apenas um único <BODY> em cada página HTML. Até a versão HTML 4.01 possui os seguintes parâmetros opcionais de configuração de cores da página web:

- **bgcolor** = cor do fundo. Padrão é branco.
- **text** = cor das fontes da página inteira. Padrão é preto.
- **link** = cor dos links. Padrão é azul.
- **vlink** = cor dos links visitados. Padrão é marrom.
- **alink** = cor dos links em acesso. Padrão é azul.

```
<BODY bgcolor="lightblue" text="darkblue"
link="red" vlink="orange" alink="pink">
```

No HTML 5 é necessário utilizar <style> em CSS.

```
<HEAD>
<style>
    a:link {
        color: red;      //para link não visitado
    }
    a:visited {
        color: green;    //para link visitado
    }
    a:hover {
        color: hotpink;  //ao passar o mouse sobre
    }
    a:active {
        color: blue;     //ao clicar sobre o link
    }
</style>
</HEAD>
```

- **background** = imagem no fundo, que pode ser multiplicada na tela. Não deve ser utilizada junto com **bgcolor**.

```
<BODY background="foto1.jpg" text="green"
link="brown" vlink="cyan" alink="magenta">
```

No HTML 5 é necessário utilizar <style> em CSS.

```
<HEAD>
<style>
    body {
        background-image:      url("foto1.jpg");
        background-repeat:     no-repeat;
        background-position:    right top;
        background-attachment: fixed;
    }
</style>
</HEAD>
```

Ou ainda, CSS no modo curto:

```
<HEAD>
<style>
    body {
        background: #00ff00 url("smiley.gif") no-repeat
        fixed center;
    }
</style>
</HEAD>
```

f. <!-- comentários -->

Permite digitar comentários em qualquer parte do programa. Esses comentários não serão exibidos na página.

```
<!-- Elaborado pelo Prof. Celso Gallão -->
```

4 - Parágrafos

O navegador, ao executar o HTML, não reconhece o fim de um parágrafo apenas porque o programador escreveu o código em linhas diferentes. Mesmo que o programador tenha pressionado [ENTER] no código fonte HTML, o navegador não reconhecerá a quebra de linha ou parágrafo. É necessário utilizar tags específicas, a saber:

a. <P>

Ao utilizar apenas <P>, especifica a quebra de parágrafo e **insere** uma linha em branco após.

```
<BODY>
Texto do 1º parágrafo <P>
Texto do 2º parágrafo
</BODY>
```

b. <P> ... </P>

Pode ser utilizado com o seu par, marcando o início e o fim de um parágrafo. Até a versão HTML 4.01 possui o seguinte parâmetro opcional:

- **align** = alinhamento (left, center, right)

```
<BODY>
<P align="right">Este parágrafo será exibido
centralizado na tela.</P>
</BODY>
```

**c.
**

Especifica uma simples quebra de linha **sem inserir** uma linha em branco após.

```
<BODY>
Texto na 1ª linha <BR>
Texto na 2ª linha
</BODY>
```

d. <PRE> ... </PRE>

Permite que um texto seja reconhecido pelo navegador com sua formatação original, reconhecendo a tecla [TAB] para tabulação e [ENTER] para final de linha. A fonte utilizada é a **Courier** pois mantém cada caracter com a mesma largura.

```
<PRE>
Agora sim,
o HTML irá reconhecer a tecla [ENTER]
```

para quebra de linha.

Até as margens com [TAB] são reconhecidas.

</PRE>

e. <DIV> ... </DIV>

Marca o início e o fim de um bloco de texto. Utilizada para agrupar e formatar elementos em um bloco. Por padrão, os navegadores inserem quebra de linha antes e após o <DIV>, mas isso pode ser alterado com CSS. Até a versão HTML 4.01 possui o seguinte parâmetro opcional:

- **align** = alinhamento (left, center, right, justify)

<DIV align="right">

Textos do bloco que serão visualizados à direita.

</DIV>

No HTML 5 é necessário utilizar <style> em CSS.

<DIV style="color:#0000FF">

Textos do bloco que serão exibidos com a cor azul.

</DIV>

f. **Textos em Cabeçalhos** <H1> até <H6>

É um tipo de formatação de texto que pode ser utilizada em qualquer lugar da página, com 6 configurações diferentes. Por padrão, são 6 tamanhos de fonte, sendo a maior com <H1> até a menor com <H6>. Automaticamente será adicionada uma linha em branco antes e outra depois. Até a versão HTML 4.01 possui o seguinte parâmetro opcional:

- **align** = alinhamento (left, center, right, justify)

<H1 align = "right">Texto Centralizado</H1>

No HTML 5 é necessário utilizar <style> em CSS.

<HEAD>

<style>

```
h1 {
    display:      block;
    font-size:    2;
    margin-top:   0.67;
    margin-bottom: 0.67;
    margin-left:  0;
    margin-right: 0;
    font-weight:  bold;
}
```

</style>

</HEAD>

g. <HR>

Até a versão HTML 4.01, <HR> traça uma linha horizontal na tela e possui os seguintes parâmetros opcionais:

- **size** = espessura da linha.
- **width** = comprimento da linha.
- **align** = alinhamento (left, center, right).
- **color** = cor da linha.

<HR SIZE=2 WIDTH=50 ALIGN="left" COLOR="red">

No HTML 5 é necessário utilizar <style> em CSS.

<HEAD>

<style>

```
hr {
    display:      block;
    margin-top:   0.5;
    margin-bottom: 0.5;
    margin-left:  auto;
    margin-right: auto;
    border-style:  inset;
    border-width:  1px;
}
```

</style>

</HEAD>

5 - Configurando Textos

As tags que alteram as configurações de textos devem ser especificadas marcando o início e o fim da aplicação. Até a versão HTML 4.01 as tags eram:

a.		texto em negrito	
b.	<I>	texto em itálico	</I>
c.	<U>	texto sublinhado	</U>
d.	⊗ <BIG>	aumenta a fonte + negrito	</BIG>
e.	<SMALL>	reduz a fonte	</SMALL>
f.	^{	texto sobrescrito	}
g.	_{	texto subscrito	}

No HTML 5 opcionalmente pode-se utilizar <style> em CSS para configurar os parâmetros, conforme segue:

<HEAD>

<style>

```
b {
    font-weight: bold;
}
i {
    font-style: italic;
}
u {
    text-decoration: underline;
}
small {
    font-size: smaller;
}
sub {
    vertical-align: sub;
    font-size: smaller;
}
sup {
    vertical-align: super;
    font-size: smaller;
}
}
```

</style>

</HEAD>

No HTML 5, todas as *tags* anteriores (exceto <BIG>) são aceitas. A seguir, uma série de *tags* que já existiam no HTML 4.01, mas que só têm sentido se configuradas com <style> em CSS:

h.		texto destacado	
i.		texto enfatizado	
j.	<MARK>	texto realçado (pintado)	</MARK>
k.	<CITE>	texto de citação	</CITE>
l.	<DFN>	texto de definição	</DFN>

À seguir exemplos da configuração padrão destas tags:


```

<HEAD>
<style>
strong {
    font-weight: bold;
}
em {
    font-style: italic;
}
mark {
    background-color: yellow;
    color: black;
}
cite {
    font-style: italic;
}
dfn {
    font-style: italic;
}
</style>
</HEAD>

```

Editar o EXERCÍCIO 01 do Caderno de Exercícios

PARTE 2 - Links e Imagens

6 - Documentos com Hiperlinks

São documentos que se vinculam a outros documentos através de ligações que podem ser textuais ou mesmo por imagens. São os chamados **LINKS**, ou seja, ligações de uma página web com outra.

6.1 - Link para Página Web Local

Pode ser texto ou imagem que, ao ser clicada, fará ligação com o documento que for mencionado, bastando para isso relacionar o endereço local do documento ao *hiperlink*. A tag utilizada é **<A>...** com o parâmetro **href**:

```
<A href="path\pagina.htm">hiperlink</A>
```

```
<A href="index.htm">Página Principal</A>
```

Quando este *link* for clicado é exibido o documento (neste caso uma página web) chamado **index.htm**, que deverá estar no mesmo diretório da página web que o acessou, caso contrário deverá ser indicado o *path* (caminho) completo do documento, com as barras invertidas "\".

```
<A href="c:\arquivos\index.htm">Voltar à Página Principal</A>
```

Para subir um nível na hierarquia dos diretórios (pastas) utilize a marcação com ponto-ponto-barra "...\".

```
<A href="..\arquivos\pagina2.htm">Página 2</A>
```

6.2 - Link para Página Web Externa

Neste caso o *link* fará ligação com um documento externo, ou seja, que está conectado a uma rede de computadores como internet ou intranet. A tag utilizada é **<A>...** com o parâmetro **href**. Entretanto, o endereço tem de ser a URL completa do documento de destino:

```
<A href="URL">hiperlink</A>
```

```
<A href="http://www.fatecsaocaetano.edu.br/">Site da Fatec São Caetano</A>
```

6.3 - Definindo onde Abrir a Página Web

A tag **<A>** possui diversos parâmetros que serão abordados mais adiante. Entretanto um deles é o responsável por informar onde a página web de destino do *link* deverá ser exibida. O parâmetro chama-se **target** e permite direcionar para **_blank**, **_parent**, **_self**, **_top** ou ainda para um nome de frame.

```
<A href="http://www.google.com" target="_blank">Site do Google</A>
```

No HTML 5 pode-se utilizar **<style>** em CSS.

```

<HEAD>
<style>
a:link, a:visited {
    color:          red;
    text-decoration: underline;
    cursor:         auto;
}
</style>
</HEAD>

```

7 - Exibição de Imagens

A tag utilizada é **** juntamente com o parâmetro obrigatório **src="URL"**. Não possui tag de fechamento. A seguir, alguns parâmetros de ****.

- **src** = URL do arquivo de imagem.
- **alt** = texto alternativo sobre a imagem.
- **width** = largura da imagem (pixels ou %).
- **height** = altura da imagem (pixels ou %).
- **border** = borda da imagem.
- **align** = alinhamento em relação ao texto.

```
<IMG src="foto1.jpg" alt="Foto de Teste" width=200 border=1 align="top">
```

Para exibir uma imagem na página web é necessário que esta esteja em um dos formatos aceitos pelo navegador:

- **GIF (Graphics Interchange Format)**: É o formato padrão original dos navegadores. Permite GIF animado - que foram muito explorados no passado - e *background* transparente. Suporta apenas 256 cores (8 bits), sendo pouco recomendado imagens muito coloridas ou com texturas.
- **JPEG (Joint Photographic Engineering Group)**: É o formato mais utilizado pela sua maior capacidade de compactação. Suporta 16.777.216 cores (24 bits), sendo recomendado para fotografias digitais de alta resolução. Também é o formato padrão da maioria das máquinas digitais e celulares com câmera digital.
- **PNG (Portable Network Graphics)**: É o formato que permite imagem com alta compactação e sem perda de qualidade. Também suporta 16.777.216 cores (24 bits), permite animação e *background* transparente.

7.1 - Imagem Utilizada como Link

Para exibir uma imagem usando-a como um *link*, basta utilizar a tag **<A href>** em conjunto com ****.

```
<A href="pagina01.htm"> <IMG src="foto.jpg"> </A>
```

No exemplo acima, quando o usuário clicar em **foto.jpg** será carregado o documento **pagina01.htm**.

7.2 - Link para Enviar E-Mail

Para que o usuário envie e-mail apenas clicando em um hiperlink, utiliza-se a tag **<A href>**, mas especifica-se no URL o parâmetro **mailto**: seguido do e-mail de destino.

```
<A href="mailto:celso.gallao@fatec.sp.gov.br">Clique aqui para enviar e-mail </A>
```

No exemplo acima, quando o usuário clicar no hiperlink será carregado o aplicativo para envio de e-mail que estiver instalado na máquina, como o *Outlook* por exemplo, e o e-mail descrito em **mailto:** já estará especificado como e-mail de destino.

Também pode-se enviar outras partes da mensagem, como **Assunto**, **Corpo**, **Texto** e ainda especificar o modo de envio **cc** (com cópia) e **bcc** (com cópia oculta). Adiciona-se o símbolo ? (interrogação) após o endereço de e-mail seguido dos parâmetros abaixo:

- **subject** = assunto do e-mail
- **body** = mensagem no corpo do e-mail
- **cc** = e-mail para receber cópia
- **bcc** = e-mail para receber cópia oculta

Todos os parâmetros acima devem estar após o símbolo ? (interrogação) e separados pelo & (e comercial).

```
<A href="mailto:celso.gallao@fatec.sp.gov.br?
subject=Felicidade&body=Estou muito feliz com esta
aula&cc=celso@teste.com&bcc=gallao@teste.com">
Clique aqui para enviar e-mail </A>
```

Editar o EXERCÍCIO 04 do Caderno de Exercícios

Editar o EXERCÍCIO 05 do Caderno de Exercícios

PARTE 3 - Configurando Listas

8 - Listas

As listas facilitam a criação de itens seriais. Podem ser **Ordenadas** (com itens numéricos) ou **Não Ordenadas** (com marcadores). Ao definir sub-listas tem-se então **Listas Aninhadas**, podendo ser **Mistas** ou **não**, conforme segue:

8.1 - Listas Não Ordenadas

A tag ` ... ` define o início e o fim de listas **não ordenadas**. Cada **marcador** será exibido na posição da tag ``.

```
<UL>
    <LI>Item 1</LI>
    <LI>Item 2</LI>
    <LI>Item 3</LI>
</UL>
```

A partir da estrutura acima, será exibido:

- Item 1
- Item 2
- Item 3

8.2 - Listas Ordenadas

A tag ` ... ` define o início e o fim de listas **ordenadas**. Cada **número** será exibido na posição da tag ``.

```
<OL>
    <LI>Item 1</LI>
    <LI>Item 2</LI>
    <LI>Item 3</LI>
</OL>
```

A partir da estrutura acima, será exibido:

1. Item 1
2. Item 2
3. Item 3

8.3 - Listas Aninhadas Mistas

Basta iniciar e finalizar *tags* de lista dentro de outras *tags* de lista.

```
<OL>
    <LI>Tecnologia em Jogos Digitais </LI>
    <LI>Tecnologia em ADS </LI>
    <UL>
        <LI>Engenharia de Software </LI>
        <LI>Programação de Scripts </LI>
        <OL>
            <LI>HTML </LI>
            <LI>Javascript </LI>
            <LI>PHP </LI>
        </OL>
        <LI>Inteligência Artificial </LI>
    </UL>
    <LI>Tecnologia em Segurança da Informação </LI>
</OL>
```

A partir do exemplo acima, será exibido:

1. Tecnologia em Jogos Digitais
2. Tecnologia em ADS
 - Engenharia de Software
 - Programação de Scripts
 1. HTML
 2. Javascript
 3. PHP
 - Inteligência Artificial
3. Tecnologia em Segurança da Informação

8.4 - Complementos

Parâmetros complementares podem ser aplicados às listas, conforme segue:

- **type** = tipo de marcador (1, a, A) em ``.
- **start** = posição inicial do marcador em ``.
- **reversed** = ordem decrescente em ``.

```
<OL reversed type="A" start="3" >
    <LI>Tecnologia em Jogos Digitais </LI>
    <LI>Tecnologia em ADS </LI>
    <LI>Tecnologia em Segurança da Informação </LI>
</OL>
```

A partir do exemplo acima, será exibido:

- C. Tecnologia em Jogos Digitais
- B. Tecnologia em ADS
- A. Tecnologia em Segurança da Informação

No HTML 5 pode-se utilizar `<style>` em CSS.

```
<HEAD>
<style>
ol {
    display: block;
    list-style-type: decimal;
    margin-top: 5px;
    margin-bottom: 5px;
    margin-left: 0;
    margin-right: 0;
    padding-left: 40px;
}
ul {
    display: block;
    list-style-type: disc;
    margin-top: 5px;
    margin-bottom: 5px;
    margin-left: 0;
    margin-right: 0;
    padding-left: 40px;
}
</style>
</HEAD>
```


Pode-se definir listas sem marcadores, utilizando, as *tags* `<DL>...</DL>` como início e fim da lista, `<DT>...</DT>` para o 1º nível e `<DD>...</DD>` para o 2º nível:

```
<DL>
  <DT> Html </DT>
  <DD> Html Básico </DD>
  <DT> Aprender </DT>
</DL>
```

A partir do exemplo acima, será exibido:

```
Html
  Html Básico
  Aprender
```

[Editar o EXERCÍCIO 02 do Caderno de Exercícios](#)

[Editar o EXERCÍCIO 03 do Caderno de Exercícios](#)

PARTE 4 - Melhorando a Aparência

9 - Cores

Os navegadores possibilitam até 256 tons para cada uma das cores padrão RGB (Red-Green-Blue). Portanto, cada cor é representada por 3 dígitos, **de 000 a 255**. O HTML só reconhecerá esses dígitos se estiverem convertidos na base numérica **Hexadecimal**, ou seja, **de 00 a FF**, no formato **#RRGGBB**. Também pode-se declarar as cores especificando seus nomes em inglês (tabela RGB). A seguir, uma pequena amostra da tabela de cores RGB.

Cor	Textual em inglês	hexadecimal
Preto	black	#000000
Branco	white	#FFFFFF
Azul	blue	#0000FF
Amarelo	yellow	#FFFF00
Verde	green	#008000
Lima	lime	#00FF00
Marrom	brown	#800000
Oliva	olive	#808000
Azul celeste	aqua	#00FFFF
Lilás	fuchsia	#FF00FF
Cinza	gray	#808080
Azul naval	navy	#000080
Roxo	purple	#800080
Verde forte	teal	#008080
Prata	silver	#C0C0C0
Vermelho	red	#FF0000

10 - Formatando Parágrafos

O padrão de alinhamento dos parágrafos em HTML é à esquerda mas podemos modificá-los, assim como o tipo e a cor da fonte, utilizando alguns parâmetros, conforme segue:

10.1 - O Parâmetro align

É utilizado nas *tags* `<P>`, `<DIV>` ou `<H1>` até `<H6>`. **align** = right, center ou left.

```
<P align = "center"> texto centralizado </P>
<DIV align = "right"> texto à direita </DIV>
<H1 align = "right"> texto à direita </H1>
```

Mas atenção, porque **align** é apenas um parâmetro das *tags* acima. Jamais utilize **align** como uma *tag* separada. **Não existe a tag <align>!**

10.2 - A Tag <CENTER>...</CENTER>

Até a versão HTML 4.01 era possível centralizar textos, como um marcador de início e fim, através da *tag* `<CENTER>...</CENTER>`. Não é reconhecido pelo HTML 5.

```
<CENTER> texto centralizado </CENTER>
```

10.3 - A Tag ...

Até a versão HTML 4.01 era possível alterar o tipo, tamanho, e a cor das fontes através da *tag* `...` e seus parâmetros abaixo. Não é reconhecido pelo HTML 5.

- **face** = tipo da fonte
- **size** = tamanho da fonte
- **color** = cor da fonte

```
<FONT face="arial" size=5 color="#FF0000">
```

10.4 - Texto em Movimento

A *tag* `<MARQUEE>...</MARQUEE>` faz com que o texto fique em movimento horizontal na tela. Esta *tag* não faz parte da documentação oficial do HTML por ser considerado efeito visual, portanto deve ser gerenciado pelo CSS. Não é reconhecido por alguns navegadores. O padrão de movimento é da direita para a esquerda, mas pode ser alterado com os seus parâmetros:

- **behavior**: define a forma de rolagem do texto:
 - **scroll** – rolagem contínua em um mesmo sentido.
 - **slide** – o texto para ao chegar ao final da rolagem.
 - **alternate** – o texto desliza de um lado para o outro, sem parar.
- **direction**: define a direção de rolagem do texto:
 - **left** - para a esquerda.
 - **right** - para a direita.
- **loop**: define o número de vezes que o texto desliza pela página:
 - infinite (padrão), define uma rolagem constante.
- **scrollamount**: define a velocidade de rolagem do texto, em milissegundos.
- **scrolldelay**: define a quantidade de pixels a percorrer em cada quadro.

```
<marquee behavior = "alternate"
  direction = "right"
  loop = 5
  scrollamount = 100
  scrolldelay = 5>
texto em movimento
</marquee>
```

[Editar o EXERCÍCIO 04 do Caderno de Exercícios](#)

[Editar o EXERCÍCIO 05 do Caderno de Exercícios](#)

PARTE 5 - Âncoras

11 - Utilizando Âncoras

As âncoras são utilizadas para fazer ligações entre partes de uma página web, através da tag `<A>` com seu parâmetro `name`. Para isso, é preciso determinar um ponto de referência na página, que será o destino a ser buscado pelo *link*. Este ponto de referência é a âncora e receberá um nome que será mencionado pelo *link* que o acessará. É bastante útil para se movimentar por textos longos na página web, como nos celulares por exemplo.

11.1 - Definindo uma Âncora

Até a versão HTML 4.01 era possível definir uma âncora com a tag `<A name>...` no ponto de destino na página web. O nome deve ser definido sem o símbolo `#` e o texto entre as marcas de início e fim não aparecem na tela.

```
<a name="inicio">Início da Página</a>
```

11.2 - Link para acessar a Âncora

Utiliza-se um texto ou uma figura que, ao ser clicada, fará ligação direta com a âncora que foi definida, fazendo com que a página web “seja movida” até a âncora. A tag usada para criar um *link* é `<A href>...`. O nome da âncora deve ser referenciado com o símbolo `#`, indicando que o destino é uma âncora na página atual.

```
<a href="#inicio">
Voltar ao Início da Página</a>
```

11.3 - Link para acessar Âncora Externa

Para acessar uma âncora externa, ou seja, definida em uma outra página web, basta referenciar a URL da página de destino imediatamente antes do símbolo `#` descrito no parâmetro `href` da tag `<A>`.

```
<a href="pagina2.htm#inicio">
Voltar ao Início da Página 2</a>
```

[Editar o EXERCÍCIO 06 do Caderno de Exercícios](#)

[Editar o EXERCÍCIO 07 do Caderno de Exercícios](#)

PARTE 6 - Tabelas

12 - Utilizando Tabelas

A utilização de tabelas melhora muito a aparência de uma página web, mas requer muita atenção, paciência e testes contínuos, para que se obtenha um bom resultado. As tags são simples, mas a estruturação delas pode ser bastante complexa, dependendo da aplicação requerida:

```
<TABLE>
<CAPTION>Título da Tabela</CAPTION>
<TR>
    <TD>...</TD>
    <TH>...</TH>
</TR>
</TABLE>
```

Até a versão HTML 4.01 os parâmetros abaixo eram utilizados. Na versão HTML 5 nem todos os parâmetros podem ser utilizados, sendo necessária a formatação com estilos em CSS.

a. A tag `<TABLE>...</TABLE>` define o início e o fim de uma nova tabela, com os seguintes parâmetros:

- `align` = alinhamento horizontal (left, center, right).
- `bgcolor` = cor do fundo.
- `border` = 1 ou 0.
- `cellspacing` = espaço em pixels entre as células.
- `cellpadding` = espaço em pixels entre o conteúdo e a borda da célula.
- `width` = comprimento da tabela em pixels ou %.

No HTML 5 pode-se utilizar `<style>` em CSS.

```
<HEAD>
<style>
table {
    display: table;
    border-collapse: separate;
    border-spacing: 2px;
    border-color: gray;
}
</style>
</HEAD>
```

b. A tag `<TR>...</TR>` inclui uma linha na tabela, mas não recebe conteúdo.

- `align` = alinhamento horizontal (left, center, right).
- `valign` = alinhamento vertical (top, middle, bottom).
- `bgcolor` = cor do fundo.

No HTML 5 pode-se utilizar `<style>` em CSS.

```
<HEAD>
<style>
tr {
    display: table-row;
    vertical-align: inherit;
    border-color: inherit;
}
</style>
</HEAD>
```

c. Os conteúdos da tabela devem ser inseridos nas células que são definidas pelas tags `<TD>...</TD>` e `<TR>...</TR>`. A diferença é que `<TR>` exibe os conteúdos com alinhamento centralizado e em negrito.

- **colspan** = qtde. de colunas mescladas
- **rowspan** = qtde. de linhas mescladas
- **align** = alinhamento horizontal (left, center, right).
- **valign** = alinhamento vertical (top, middle, bottom).
- **bgcolor** = cor do fundo.
- **border** = 1 ou 0.
- **width** = comprimento da célula em pixels ou %.
- **height** = comprimento da célula em pixels ou %.
- **cellspacing** = espaço em pixels entre as células.
- **cellpadding** = espaço em pixels entre o conteúdo e a borda da célula.
- **nowrap** Se declarado, não permite quebra de texto na célula.

No HTML 5 pode-se utilizar `<style>` em CSS.

```
<HEAD>
<style>
td {
    display: table-cell;
    vertical-align: inherit;
}
</style>
</HEAD>
```

d. A tag `<CAPTION>...</CAPTION>` inclui um título que será exibido centralizado e acima da tabela. Deve ser declarado imediatamente após a tag `<TABLE>`.

- **align** = alinhamento horizontal (left, center, right).

No HTML 5 pode-se utilizar `<style>` em CSS.

```
<HEAD>
<style>
caption {
    display: table-caption;
    text-align: center;
}
</style>
</HEAD>
```

```
<TD>15,33</TD>
<TD>149.263</TD>
</TR>
</TABLE>
```

12.2 - Tabela com Células Mescladas

Times do Futebol Paulista

Palmeiras		São Paulo
Corinthians	Portuguesa	São Caetano
Santos	Santo André	

```
<TABLE border=2 width=100% bgcolor="yellow">
<CAPTION>Times do Futebol Paulista</CAPTION>
<TR>
    <TH colspan=2 bgcolor=green>Palmeiras</TH>
    <TD height=50>São Paulo</TD>
</TR>
<TR>
    <TD valign=bottom height=50>Corinthians</TD>
    <TH align=right>Portuguesa</TH>
    <TH rowspan=2>São Caetano</TH>
</TR>
<TR>
    <TD height=50 valign=top align=center>
        Santos</TD>
    <TD bgcolor=blue><font color=white>Santo
        André</font></TD>
</TR>
</TABLE>
```

[Editar o EXERCÍCIO 08 do Caderno de Exercícios](#)

[Editar o EXERCÍCIO 09 do Caderno de Exercícios](#)

12.1 - Tabela com Células Simples

ABC Paulista

Cidade	Área (Km ²)	População (2010)
Santo André	175	568.538
São Bernardo do Campo	406	736.466
São Caetano do Sul	15,33	149.263

```
<TABLE border=1 width=100%>
<caption>ABC Paulista</caption>
<TR>
    <TH>Cidade</TH>
    <TH>Área (Km<sup>2</sup></sup></TH>
    <TH>População (2010)</TH>
</TR>
<TR>
    <TD>Santo André</TD>
    <TD>175</TD>
    <TD>568.538</TD>
</TR>
<TR>
    <TD>São Bernardo do Campo</TD>
    <TD>406</TD>
    <TD>736.466</TD>
</TR>
<TR>
    <TD>São Caetano do Sul</TD>
```

PARTE 07 - Frames

13 - Utilizando Frames

A utilização de *frames* possibilita a divisão da janela do navegador em vários quadros, possibilitando o carregamento de páginas *web* diferentes em cada quadro, numa mesma janela. É possível também interagirmos em um *frame* e vermos o resultado em outro *frame*.

Até a versão HTML 4.01 era necessário utilizar um página *web* contendo apenas as configurações dos *frames*, incluindo as tags `<FRAMESET>` e `<FRAME>`. Na versão HTML5 somente pode-se utilizar a tag `<IFRAME>`.

13.1 - Definindo o layout dos Frames

Até a versão HTML 4.01 era utilizada a tag `<FRAMESET>` para definir a aparência da página dos *frames*, com os seguintes parâmetros:

- a. `<cols>` = cria *frames* dispostos em colunas. Deve-se especificar os comprimentos das colunas - separados por vírgulas - em pixels, porcentagens ou relativos.

```
<FRAMESET cols = 200,*>... </FRAMESET>
```

200px de comprimento	* comprimento que sobrar
----------------------	-----------------------------

```
<FRAMESET cols = 60%,40%>... </FRAMESET>
```

60% de comprimento	40% de comprimento
--------------------	--------------------

- b. `<rows>` = cria *frames* dispostos em linhas. Deve-se especificar a altura das linhas - separados por vírgulas - em pixels, porcentagens ou relativos.

```
<FRAMESET rows = 100,300,200>... </FRAMESET>
```

100px de altura
300px de altura
200px de altura

```
<FRAMESET rows = 200,*>... </FRAMESET>
```

200px de altura
* comprimento que sobrar

13.2 - Definindo o Conteúdo dos Frames

Até a versão HTML 4.01 era utilizada a tag `<FRAME>` para atribuir o conteúdo de cada *frame*, com os seguintes parâmetros:

- a. `<frameborder>` = define as bordas dos *frames* (1 ou 0).
b. `<marginheight>` = altura das margens superior e inferior, em pixels.
c. `<marginwidth>` = largura das margens esquerda e direita, em pixels.
d. `<name>` = define um nome ao *frame*.
e. `<noresize>` = evita que as bordas sejam redimensionadas pelo usuário.
f. `<scrolling>` = define a exibição de barras de rolagem (YES ou NO).
g. `<src>` = endereço (URL) da página web que será exibida.

```
<FRAMESET cols = 100,300,200>  
<FRAME name="frA" src="pag1.htm">...</FRAME>  
<FRAME name="frB" src="pag2.htm">...</FRAME>  
<FRAME name="frC" src="pag3.htm">...</FRAME>  
</FRAMESET>
```

100px frA pag1.htm	300px frB pag2.htm	200px frC pag3.htm
<pre><FRAMESET rows = 60%,40%> <FRAME name="frA" src="pag1.htm">...</FRAME> <FRAME name="frB" src="pag2.htm">...</FRAME> </FRAMESET></pre>		
60% frA pag1.htm		
40% frB pag2.htm		

13.3 - Definindo Frames Aninhados

É bastante comum encontrarmos páginas web utilizando *frames* aninhados, ou seja, *frames* dentro de *frames*, combinando parâmetros `<cols>` e `<rows>` de `<FRAMESET>`.

```
<HTML>  
<HEAD>  
<TITLE>...</TITLE>  
</HEAD>  
<FRAMESET rows = 100,*>  
<FRAME name = FrameA src="DELTA.htm">  
<FRAMESET cols = 200,*>  
<FRAME name = FrameB src="ALFA.htm">  
<FRAME name = FrameC src="BETA.htm">  
</FRAMESET>  
</FRAMESET>  
</BODY>
```

100px de altura FrameA DELTA.htm	
200px de comprimento FrameB ALFA.htm	comprimento que sobrar FrameC BETA.htm

Pode-se utilizar o **FrameA** para exibir propagandas (*banners*) e informações sobre o *site*; o **FrameB** utilizado para abrigar os *links*, e o **FrameC** utilizado para receber as páginas acessadas pelos *links*.

ATENÇÃO: Na página de configuração dos *frames* não pode-se utilizar a tag `<BODY>`, caso contrário as tags `<FRAMESET>` e `<FRAME>` não farão efeito.

13.4 - Carregando Páginas em outro Frame

Por padrão, quando o usuário clica em um link a página web é carregada no mesmo frame da página aonde está o link, entretanto, isso pode ser configurado. Para adirecionar o carregamento de uma página web para um determinado frame, pode-se utilizar duas técnicas, a saber:

- a. A tag `<BASE target="destino">`, declarada na área do `<HEAD>`, define um *frame* padrão para todos os *links*, ou seja, todos os *links* desta página web irão direcionar o carregamento para o frame de destino definido pelo parâmetro `target`.

```
<HEAD>  
<BASE target="FrameB">  
</HEAD>
```

- b. O parâmetro `target` da tag `<A>...` define o destino do carregamento da página especificamente para este *link*.

```
<A href="GAMA.htm" target="FrameC">
```

13.5 - Utilizando <IFRAME>...</IFRAME>

A tag <IFRAME> especifica um quadro *inline*, utilizado para incorporar outra página web, como um *frame*. Não precisa de uma página de configuração de *frames*, sendo aplicado diretamente no local desejado, na própria página web.

- a. **name** = define o nome do quadro.
- b. **src** = URL da página web que será exibida.
- c. **width** = comprimento do quadro, em pixels.
- d. **height** = altura do quadro, em pixels.

```
<iframe name="frB" width=200 height=300  
src="ALFA.htm"></iframe>
```

13.6 - Definindo <IFRAME> em <DIV>

Atualmente, utilizam-se frames em páginas web utilizando <IFRAME> dentro de áreas delimitadas pela tag <DIV>, organizando os quadros em um único documento HTML.

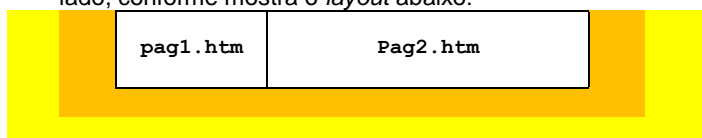
Deve-se definir em CSS os elementos utilizados para configurar o <DIV>. O exemplo abaixo mostra 2 elementos do tipo #id:

```
<style>  
#headerwrapper {  
    background: yellow;  
    width: 100%;  
    height: 400px;  
}  
#header {  
    width: 1000px;  
    height: 300px;  
    background: orange;  
    margin: 0 auto;  
    text-align: center;  
}  
</style>
```

Deve-se definir, então, as tags <IFRAME> dentro das tags <DIV> com os elementos #id. O exemplo abaixo mostra 2 frames dispostos lado a lado:

```
<div id="headerwrapper">  
    <div id="header">  
        <iframe name="frame1" width=200 height=200  
            src="pag1.htm"></iframe>  
        <iframe name="frame2" width=580 height=200  
            src="pag2.htm"></iframe>  
    </div>  
</div>
```

Como resultado, serão exibidas as 2 páginas web lado a lado, conforme mostra o layout abaixo:



[Editar o EXERCÍCIO 10 do Caderno de Exercícios](#)

[Editar o EXERCÍCIO 11 do Caderno de Exercícios](#)

PARTE 08 - Formulários

14 - Utilizando Formulários

Os formulários eletrônicos gerados pelo HTML são recursos essenciais para a troca de informações entre usuário e servidor. Formulários enviam dados que serão processados no servidor e posteriormente devolvidos ao cliente.

Os exemplos aqui mostrados funcionarão para efeito de teste, pois para que um formulário troque informações com o servidor é necessário acessar sua página web através de um servidor, além de utilizar páginas para fazer o processamento dos dados escritas em linguagens dinâmicas de programação como PHP por exemplo.

Também pode-se utilizar a linguagem de programação Javascript para fazer validação dos campos de formulário, antes do envio dos dados ao servidor.

14.1 - A Tag <FORM>...</FORM>

Possui funções importantes como especificar o local da página que controlará o formulário, especificar a forma como os dados serão enviados e qual a página receberá os dados enviados. Utiliza os seguintes parâmetros:

- a. **name** = define o nome do formulário.
- b. **action** = especifica o URL e a página no servidor que irá processar os dados enviados pelo formulário.
- c. **method** = indica o método usado para envio dos dados: **POST** ou **GET** (default).

```
<form name="form" action="pag_dados.php"  
method="POST">
```

Quando GET é usado, os dados enviados são visíveis no campo de endereço do navegador, ficam visíveis anexados ao link. **GET não deve ser usado para enviar informações confidenciais!** O método GET é mais adequado para quantidades curtas de dados não sigilosos, e também possui limitações de quantidade de caracteres.

Use sempre POST se os dados do formulário contiver informações sigilosas ou pessoais. O método POST não exhibe os dados do formulário no campo de endereço do navegador. POST não possui limitações de quantidade de caracteres podendo ser usado no envio de grandes quantidades de dados.

14.2 - A Tag <INPUT>

Define campos para entrada de dados a partir do usuário da página web, com os seguintes parâmetros e atributos:

- a. **type** = especifica o tipo do elemento a ser configurado, podendo ser:

Elementos	Aplicação
TEXT	- caracteres.
PASSWORD	- senha oculta, exibe asteriscos.
CHECKBOX	- caixa de verificação (sim/não).
RADIO	- botão de rádio (exclusivo).
SUBMIT	- enviar o conteúdo do formulário.
RESET	- apagar o conteúdo do formulário.
BUTTON	- botão genérico.
HIDDEN	- elemento e conteúdo ocultos.

- b. **name** = todo objeto precisa ter seu próprio nome, exceto para os tipos CHECKBOX e RADIO onde todos os itens de um grupo possuem o mesmo nome.
- c. **value** = depende do tipo do objeto:

Elementos	Aplicação de Value
TEXT ou PASSWORD	- conteúdo digitado.
CHECKBOX ou RADIO	- valor a ser enviado.
SUBMIT, RESET, ou BUTTON	- rótulo do botão.

- d. size** = especifica o tamanho da caixa de texto utilizada pelos objetos **TEXT** ou **PASSWORD**. Tamanho *default* é 20.
- e. maxlength** = especifica a quantidade máxima de caracteres aceitos para digitação nos objetos **TEXT** ou **PASSWORD**.
- f. checked** = atributo que determina o valor *default* em um objeto do tipo **CHECKBOX** ou **RADIO**.
- g. disabled** = atributo que determina que o elemento está desabilitado.
- h. readonly** = atributo que determina que o elemento é apenas para leitura, não permitindo alteração.
- **Utilizando TEXT e PASSWORD:** a diferença é que no elemento **PASSWORD** asteriscos são exibidos no lugar dos caracteres digitados:

Login do usuário:

```
<INPUT type="TEXT" name="senha_nova" value="Joaozinho" disabled><P>
```

Digite o nome do aluno:

```
<INPUT type="TEXT" name="nome_aluno" value="" size=30 maxlength=40><P>
```

Digite a nova senha do aluno:

```
<INPUT type="PASSWORD" name="senha_nova" value="" size=8 maxlength=8><P>
```

Nome do curso:

```
<INPUT type="TEXT" name="nome_curso" value="ADS" readonly>
```

Login do usuário:

Digite o nome do aluno:

Digite a nova senha do aluno:

Nome do curso:

- **Utilizando CHECKBOX e RADIO:** a diferença é que no elemento **CHECKBOX** é permitida a marcação de múltiplas opções, e o nome do elemento **CHECKBOX** deve-se indicar os caracteres `[]` para informar ao navegador que trata-se de um vetor que pode possuir diversos valores marcados pelo usuário. Entretanto no nome do elemento **RADIO** não se indica como vetor pois possuirá apenas 1 valor escolhido pelo usuário:

Marque os sabores de sorvete que mais gosta:


```
<INPUT type="CHECKBOX" name="sabor[]" value="morango">Morango<BR>
```

```
<INPUT type="CHECKBOX" name="sabor[]" value="chocolate" checked>Chocolate<BR>
```

```
<INPUT type="CHECKBOX" name="sabor[]" value="creme" disabled>Creme<BR>
```

```
<INPUT type="CHECKBOX" name="sabor[]" value="flocos">Flocos<BR>
```

Marque os sabores de sorvete que mais gosta:

☐ Morango

☒ Chocolate

☐ Creme

☐ Flocos

Escolha a sua marca preferida de sorvete:


```
<INPUT type="RADIO" name="marca" value="kibon">Kibon<BR>
```

```
<INPUT type="RADIO" name="marca" value="nestle" checked>Nestlé<BR>
```

```
<INPUT type="RADIO" name="marca" value="copenhague" disabled>Copenhague<BR>
```

```
<INPUT type="RADIO" name="marca" value="bariloche">Bariloche<BR>
```

```
value="bariloche">Bariloche<BR>
```

Escolha a sua marca preferida de sorvete:

☐ Kibon

☒ Nestlé

☐ Copenhagen

☐ Bariloche

- **Utilizando RESET, SUBMIT e BUTTON:** a diferença é que o elemento **RESET** apenas apaga todos os campos do formulário restabelecendo os valores *default*, enquanto que o elemento **SUBMIT** envia todos os dados do formulário de acordo com as configurações de **method** e **action** declaradas na **tag <FORM>**. Já o elemento **BUTTON**, apenas cria um botão genérico sem função prédefinida, ou seja, necessita de uma configuração extra (em javascript, por exemplo) para ter alguma funcionalidade:

```
<INPUT type="RESET" name="botao_limpar" value="APAGAR DADOS">
```

```
<INPUT type="SUBMIT" name="botao_enviar" value="ENVIAR DADOS">
```

```
<INPUT type="BUTTON" name="botao_generico" value="VALIDAR DADOS" onClick="validar()">
```

• A Tag <INPUT> em HTML 5

No HTML 5, a **tag <INPUT>** recebeu novos elementos e parâmetros. A seguir, estão descritas algumas destas novidades, entretanto podem não ser aceitas por alguns navegadores ou versões mais antigas, como por exemplo o IE9 e inferiores:

- a. type** = especifica o tipo do elemento HTML 5:

Elementos	Aplicação
NUMBER	- apenas números.
RANGE	- exibe uma barra de valores com controle deslizante, com <i>default</i> de 0 a 100.
FILE	- permite selecionar arquivos na estrutura de pastas.

- b. max** = especifica valor máximo para a entrada.
- c. min** = especifica valor mínimo para a entrada.
- d. form** = especifica que o elemento faz parte de um formulário. Apenas aplicado em elementos declarados fora da **tag <FORM> . . . </FORM>**.
- e. required** = atributo que define que o elemento possui entrada obrigatória, não podendo estar em branco. Esta validação ocorre apenas no envio do formulário.
- f. autofocus** = atributo que define que o elemento receberá o foco quando a página for carregada.

Digite a quantidade de sorvetes (0-10):

```
<INPUT type="NUMBER" name="qtd_sorvete" value="" min=0 max=10 required>
```

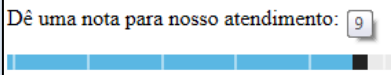
Digite a quantidade de sorvetes (0-10):

Este é um campo obrigatório

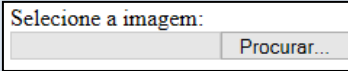
Digite a quantidade de sorvetes (0-10):

Você deve inserir um valor entre 0 e 10

Dê uma nota para nosso atendimento:
<input type="RANGE" name="nota" min=0 max=10>



Selecione a imagem:
<input type="FILE" name="img" multiple>



• A Tag <LABEL>...</LABEL> em HTML 5

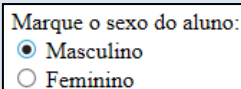
Define um rótulo para um elemento <INPUT>, com a vantagem de tornar este rótulo clicável ao usuário, alternando o controle do elemento ao clique do mouse sobre o rótulo, facilitando sua usabilidade:

- for** = especifica a qual elemento <INPUT> o rótulo está vinculado. Deve ser igual ao atributo **id** do <INPUT> relacionado.
- form** = especifica que o elemento faz parte de um formulário. Para elementos declarados fora da tag <FORM>...</FORM>.

Marque o sexo do aluno:

<INPUT type="RADIO" name="sexo"
id="male" value="M">
<LABEL for="male">Masculino</LABEL>

<INPUT type="RADIO" name="sexo"
id="female" value="F">
<LABEL for="female">Feminino</LABEL>



14.3 - A Tag <SELECT>...</SELECT>

Define uma lista *drop-down* contendo vários itens. Cada item é definido por uma tag <OPTION>...</OPTION>:

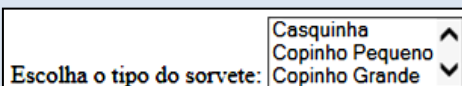
- name** = especifica o nome da lista.
- size** = especifica a quantidade de itens exibidos, determinando o espaço ocupado na tela.
- multiple** = atributo que determina que vários itens podem ser selecionados simultaneamente.

• Lista com <OPTION>...</OPTION>

Define um item da lista suspensa (*drop-down*) para que o usuário selecione um item. Por padrão, o 1º item será selecionado, ou aquele que receber o atributo **selected**:

- value** = especifica o conteúdo de cada item, a ser enviado pelo formulário.
- selected** = atributo que define o item *default*.

Escolha o tipo do sorvete:
<SELECT name="tipo_sorvete" size=3>
<OPTION value="Casquinha">Casquinha</OPTION>
<OPTION value="Copinho Pequeno">Copinho Pequeno</OPTION>
<OPTION value="Copinho Grande">Copinho Grande</OPTION>
<OPTION value="Big Master">Big Master</OPTION>
</SELECT>



14.4 - A Tag <DATALIST>...</DATALIST>

No HTML 5, a tag <DATALIST> define uma lista suspensa (*drop-down*) contendo vários itens. Cada item é definido por uma tag <OPTION>. Mas antes, é necessária a declaração de um elemento <INPUT list="">. O parâmetro **list** do elemento <INPUT> deve referir-se ao atributo **id** do elemento <DATALIST>.

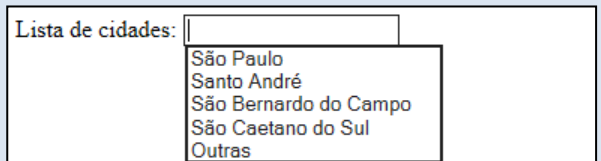
- id** = especifica o nome da lista.

• Lista com <OPTION> em HTML 5

Define um item da lista suspensa (*drop-down*) HTML 5, para que o usuário selecione um item. Por padrão, o 1º item será selecionado, ou aquele que receber o atributo **selected**. Na versão HTML 5, não precisa fechar cada opção com </OPTION>, além disso, o conteúdo do parâmetro **value** já é mostrado na lista como um rótulo da opção:

- value** = especifica o conteúdo de cada item, a ser enviado pelo formulário.
- selected** = atributo que define o item *default*.

Lista de cidades:
<INPUT list="cidades" name="lista_cidades">
<DATALIST id="cidades">
<OPTION value="São Paulo">
<OPTION value="Santo André">
<OPTION value="São Bernardo do Campo">
<OPTION value="São Caetano do Sul" selected>
<OPTION value="Outras">
</DATALIST>



14.5 - A Tag <TEXTAREA>...</TEXTAREA>

Define uma área para digitação de textos, com quantidade de linhas e colunas pré-definidos com os seguintes parâmetros:

- name** = especifica o nome da área de texto.
- rows** = especifica a quantidade de linhas que serão exibidas na área de texto.
- cols** = especifica quantidade de colunas que serão exibidas na área de texto.
- disabled** = atributo que determina que o elemento está desabilitado.
- readonly** = atributo que determina que o elemento é apenas para leitura, não permitindo alteração.

• A Tag <TEXTAREA> em HTML 5

No HTML 5, a tag <TEXTAREA> recebeu novos elementos e parâmetros. A seguir, estão descritas algumas destas novidades, entretanto podem não ser aceitas por alguns navegadores ou versões mais antigas, como por exemplo o IE9 e inferiores:

- form** = especifica que o elemento faz parte de um formulário. Apenas aplicado em elementos declarados fora da tag <FORM>...</FORM>.
- maxlength** = especifica a quantidade máxima de caracteres aceitos para digitação.

- c. **placeholder**=especifica uma pequena dica sobre o valor esperado, mas é apagada automaticamente quando recebe digitação.
- d. **wrap** = especifica como o texto deve ser envolvido na entrega do formulário, com ou sem quebra de linhas: (**hard** ou **soft**). No caso de entregar com quebra de linha (**hard**) tem que declarar o parâmetro **cols**.
- e. **required** = atributo que define que o elemento possui entrada obrigatória, não podendo estar em branco. Esta validação ocorre apenas no envio do formulário.
- f. **autofocus** = atributo que define que o elemento receberá o foco quando a página for carregada.

Escreva um comentário:


```
<TEXTAREA name="area_de_texto" rows=3 cols=40
placeholder="máximo de 150 caracteres..."
maxlength=150 wrap="hard" required>
</TEXTAREA>
```

Escreva um comentário:

máximo de 150 caracteres...

14.6 - A Tag <BUTTON>...</BUTTON>

Cria um botão clicável em um formulário. A diferença de <BUTTON> em relação a <INPUT type="BUTTON"> é que permite inserir imagens e configurar tipos diferentes.

- a. **type** = especifica o tipo do botão a ser configurado: (**submit**, **reset** ou **button**)
- b. **name** = especifica o nome do botão.
- c. **value** = especifica um valor para o botão.

A partir do HTML 5 foram incluídos diversos parâmetros. A seguir estão algumas das novidades:

- a. **form** = especifica que o elemento faz parte de um formulário. Para elementos declarados fora da tag <FORM>...</FORM>.
- b. **disabled** = atributo que determina que o elemento está desabilitado.
- c. **autofocus**= atributo que define que o elemento receberá o foco qdo. a página for carregada.

```
<BUTTON type="button" class="btn tipo1">
  Prog. de Scripts</BUTTON>
<BUTTON type="button" class="btn tipo2">
  Prog. O.O.</BUTTON>
<BUTTON type="button" class="btn tipo3">
  Inglês</BUTTON>
```

Prog. de Scripts

Prog. O.O.

Inglês

```
<STYLE>
.btn {
  border: none;
  color: white;
  padding: 14px 28px;
  font-size: 16px;
  cursor: pointer;
}
.tipo1 {background-color: darkgreen;}
.tipo1:hover {background-color: green;}
.tipo2 {background-color: navy;}
.tipo2:hover {background-color: blue;}
.tipo3 {background-color: #ff9800;}
.tipo3:hover {background-color: #e68a00;}
</STYLE>
```

14.7 - A Tag <FIELDSET>...</FIELDSET>

Desenha uma moldura ao redor dos elementos agrupados em um formulário. Permite a inserção de uma legenda com a tag <LEGEND> e uma tecla de atalho com o parâmetro **accesskey**. A partir do HTML 5 foram incluídos os seguintes parâmetros:

- d. **name** = especifica o nome do <FIELDSET>.
- e. **form** = especifica que o elemento faz parte de um formulário. Apenas aplicado em elementos declarados fora da tag <FORM>...</FORM>.
- f. **disabled** = atributo que determina que o elemento está desabilitado.

• A Tag <LEGEND>...</LEGEND>:

Especifica uma legenda para o grupo de elementos de formulário. Pode-se especificar uma tecla de atalho com o parâmetro **accesskey**. Deve-se declarar entre as tags <FIELDSET>...</FIELDSET>

• O Parâmetro **accesskey="tecla">**

No HTML 5, **accesskey** especifica uma tecla de atalho para mover o foco ao elemento. Pode-se utilizar em qualquer elemento HTML, entretanto, há diferenças consideráveis que variam conforme o navegador:

Navegador	Windows	Linux	Mac
Internet Explorer	[Alt] + tecla	N/A	N/A
Chrome	[Alt] + tecla	[Alt] + tecla	[Control] [Alt] + tecla
Firefox	[Alt] [Shift] + tecla	[Alt] [Shift] + tecla	[Control] [Alt] + tecla
Safari	[Alt] + tecla	N/A	[Control] [Alt] + tecla
Opera	Opera 15 ou acima: [Alt] + tecla Opera 12.1 ou abaixo: [Shift] [Esc] + tecla	N/A	N/A

```
<FIELDSET name="box1" form="F1">
  <LEGEND accesskey="a">Dados do aluno [a]
</LEGEND>
  Digite o nome do aluno:
  <INPUT type="TEXT" name="nome_aluno"><P>
  Digite a nova senha do aluno:
  <INPUT type="PASSWORD" name="senha_nova">
</FIELDSET>
```

Dados do aluno [a]

Digite o nome do aluno:

Digite a nova senha do aluno:

[Editar o EXERCÍCIO 12 do Caderno de Exercícios](#)

[Editar o EXERCÍCIO 13 do Caderno de Exercícios](#)

15 - Definindo Metadados

Metadados são dados de alto nível numa página web. A tag **<META>** fornece metadados sobre o documento HTML. Os metadados não são exibidos na página, mas sim analisados pelo navegador, sendo responsável por importantes aspectos como a criação de documentos dinâmicos ou informações específicas que serão usadas pelo servidor ou pelos mecanismos de busca da internet, como por exemplo exibir conteúdo ou recarregar a página, auxiliar os motores de busca por palavras-chave ou outros serviços da web.

15.1 - A Tag <META>

Os elementos **<META>** normalmente são usados para especificar a descrição da página, palavras-chave, autor, a última modificação e outras metainformações da página web. A tag **<META>** deve sempre ser declarada na área de cabeçalho, ou seja, dentro de **<HEAD>...</HEAD>**. A sintaxe básica é:

```
<META http-equiv=resposta content=descrição
      name=descrição url=endereço>
```

a. http-equiv define informações nos cabeçalhos dos documentos, podendo ser:

- **refresh** recarrega a página web, ou uma outra indicada pelo parâmetro **url**.
- **content-type** define o conjunto de caracteres e o tipo de página.
- **default-style** define a folha de estilo padrão.

```
<HEAD>
<META http-equiv="refresh"
      content="30, url=pagina2.htm">
<META http-equiv="content-type"
      content="text/html; charset=UTF-8">
<META http-equiv="default-style"
      content="Folha de estilo preferida">
</HEAD>
```

b. name nome da metainformação ou variável que está sendo criada, e que deve ser atribuída a um conteúdo através de **content**, podendo ser:

- **application-name** define o nome para a página.
- **author** permite definir o autor da página.
- **description** permite descrever brevemente o conteúdo da página.
- **generator** permite descrever o *framework* que gerou a página.
- **keywords** define algumas palavras-chave para auxiliar os motores de busca.

```
<HEAD>
<META name="application-name"
      content="Site de HTML">
<META name="description"
      content="Apostila de HTML">
<META name="keywords"
      content="HTML5, CSS">
<META name="author"
      content="Celso Gallão">
<META name="generator"
      content="FrontPage 4.0">
</HEAD>
```

c. content define a informação associada ao nome da variável ou metainformação.

d. url define a URL do documento.

15.2 - A Tag <META> no HTML 5

No HTML 5, o desenvolvedor passa a ter o controle *viewport*, através da tag **<META name="viewport">**. A *viewport* é a área visível do usuário de uma página web, que varia conforme o dispositivo, sendo menor em um *smartphone* do que em um *desktop*, por exemplo. Deve-se incluir o que segue em todas as páginas web do site:

```
<META name="viewport"
      content="width=device-width, initial-scale=1.0">
```

O *viewport* declarado na tag **<META>** fornece instruções do navegador sobre como controlar as dimensões e a escala da página, para a criação de páginas responsivas, que se adaptam automaticamente ao dispositivo onde estão sendo exibidas. O parâmetro **width = device-width** define que a largura da página seguirá a largura da tela do dispositivo, enquanto que o parâmetro **initial-scale = 1.0** define o nível de *zoom* inicial quando a página é carregada pela primeira vez pelo navegador.

O HTML 5 também inseriu um novo parâmetro:

e. charset define a codificação de caracteres para o documento HTML, podendo ser:

- **UTF-8** caracteres para Unicode.
- **ISO-8859-1** caracteres para o alfabeto latino.

```
<HEAD>
<META charset="UTF-8">
</HEAD>
```

Editar o EXERCÍCIO 14 do Caderno de Exercícios

16 - Introdução ao CSS

As folhas de estilo em cascata (*Cascading Style Sheets*) são utilizadas para configurar a apresentação visual da página web e agilizar a manutenção de um site, através da centralização das configurações de uma página - ou diversas - em um só lugar. Também aumenta as possibilidades de formatação visual das páginas pois permite alterar parâmetros e atributos de todos os elementos que compõem a página web, descrevendo como os elementos serão exibidos na tela.

Uma folha de estilo pode ser definida como um gabarito que formata os elementos HTML de acordo com as preferências determinadas pelo desenvolvedor, economizando muito trabalho na manutenção dos elementos HTML. A *World Wide Web Consortium (W3C)* criou o CSS com a intenção de remover a formatação de estilos aplicadas diretamente no HTML e que infernizavam a vida dos desenvolvedores.

16.1 - Regras de Utilização

A maneira mais simples de utilizar a folha de estilo é através da tag `<STYLE>...</STYLE>`, que é declarada dentro da seção `<HEAD>`. Cada regra de estilo é composta por um SELETOR, que é um elemento HTML que receberá o estilo, como `<BODY>`, `<P>`, `<H1>` por exemplo. As tags HTML têm, assim, suas propriedades alteradas pelos seus valores de atributos. Sua sintaxe básica é:

```
SELETOR {
    propriedade1: valor;
    propriedade2: valor;
}
```

O SELETOR aponta para o elemento HTML que a ser configurado. Cada bloco de SELETOR pode conter uma ou mais declarações separadas por ponto e vírgula. Cada declaração inclui um nome de propriedade CSS e um valor, separados por dois pontos. Uma declaração de CSS sempre termina com um ponto-e-vírgula, e os blocos de declaração são cercados por chaves.

No exemplo abaixo, todos os elementos `<P>` terão fonte na cor verde e alinhamento centralizado, enquanto que os elementos `<H1>`, `<H2>` e `<H3>` terão fonte arial ou verdana (nesta ordem), na cor vermelha:

```
<HEAD>
  <STYLE>
    /* Parágrafo verde e centralizado */
    P {
        color: green;
        text-align: center;
    }

    /*
    H1, H2 e H3
    em vermelho e
    arial (ou verdana)
    */

    H1,H2,H3 {
        color: red;
        font-family: "arial, verdana";
    }
  </STYLE>
</HEAD>
```

16.2 - O Seletor id

É um seletor identificador de um elemento HTML, utilizado para selecionar um elemento específico. O `id` de um elemento deve ser exclusivo dentro de uma página, sendo definido com o caractere *hash* (#) seguido do `id` do elemento.

No exemplo abaixo, foi declarado um estilo chamado `#tipol`, que será aplicado apenas no elemento que possuir o `id="tipol"`:

```
<HEAD>
  <STYLE>
    #tipol {
        color: green;
        text-align: center;
    }
  </STYLE>
</HEAD>
<BODY>
  <P id="tipol">Olá Mundo!</P>
  <P>Este parágrafo não possui estilo.</P>
</BODY>
```

16.3 - O Seletor class

É um seletor identificador de uma classe de estilo, utilizado para selecionar uma classe específica. O seletor `class` de um elemento deve ser definido com o caractere ponto (.) seguido do nome da classe de estilo. No exemplo abaixo, todos os elementos HTML que foram declarados com o seletor `class="centro"` são exibidos com alinhamento centralizado:

```
<HEAD>
  <STYLE>
    .centro {
        text-align: center;
    }
  </STYLE>
</HEAD>
<BODY>
  <H1 class="centro">Texto centralizado</H1>
  <P class="centro">Parágrafo centralizado.</P>
</BODY>
```

Pode-se declarar uma classe específica de um elemento HTML, sendo que esta classe somente afetará o elemento declarado e nenhum outro. No exemplo abaixo a classe `class="centro"` foi declarada como exclusiva do elemento `<P>`:

```
<HEAD>
  <STYLE>
    P.centro {
        text-align: center;
    }
  </STYLE>
</HEAD>
<BODY>
  <H1 class="centro">Texto não afetado</H1>
  <P class="centro">Parágrafo centralizado.</P>
</BODY>
```

Pode-se também **aplicar mais de uma classe** de estilos nos elementos HTML, conforme o exemplo abaixo:

```
<HEAD>
  <STYLE>
    P.centro {
      text-align: center;
    }
    .grande {
      font-size: 150%;
    }
  </STYLE>
</HEAD>
<BODY>
  <H1 class="grande">Texto aumentado 150%</H1>
  <P class="centro grande">Parágrafo
    centralizado e aumentado 150%</P>
</BODY>
```

17 - Aplicando Estilos

Dentre as maneiras de se aplicar folhas de estilo, cada uma delas apresenta vantagens e desvantagens. Vamos aqui conhecer as três formas mais utilizadas:

17.1 - O Método *EMBEDDING*: <STYLE>

(Incorporando) - consiste em declarar os estilos dentro do documento HTML, na seção <HEAD>. Essa é a técnica mais simples de ser utilizada, como pode-se ver nos exemplos exibidos até aqui. Este método tem como vantagem configurar individualmente cada página do site, sendo que a incorporação é feita através do conjunto de tags <STYLE>...</STYLE>:

```
<STYLE type="text/css">
  SELETOR {
    propriedade1: valor;
    propriedade2: valor;
  }
</STYLE>
```

17.2 - O Método *INLINE*: <Tag style>

(na linha) - consiste em declarar os estilos aplicando diretamente no elemento HTML, ou seja, nos próprios seletores. Essa técnica tem como vantagem configurar apenas uma ocorrência do estilo, ou para anular somente naquele elemento um estilo pré configurado. A declaração é feita através do parâmetro **style** dentro da *tag* do seletor na ocorrência desejada.

<Tag style = seletor: valor>

```
<BODY>
  <P style = border: "none">
</BODY>
```

17.3 - O Método *LINKING*: <LINK>

(Ligando) - consiste em carregar as configurações de estilo que foram descritas em um arquivo externo com extensão CSS. Essa técnica tem como vantagem a facilidade de alterar os estilos de todo o site, ou seja, de várias páginas, de uma só vez. Para tanto, deve-se declarar a *tag* <LINK> em todas as páginas que sofrerão as alterações deste arquivo externo. Veja a sintaxe, que deve estar sempre na seção <HEAD>:

<LINK rel="stylesheet" href="arquivo.css" type="text/css">

- a. **rel** define a relação entre o documento atual e o documento vinculado.
- b. **href** define a localização do documento vinculado.
- c. **type** define o tipo de mídia do documento vinculado.

```
<HEAD>
  <LINK rel="stylesheet" href="exemplo.css"
    type="text/css">
</HEAD>
```

17.4 - Unidades de Medidas

As unidades de medida utilizadas na configuração de estilos dependem do seletor, mas podem ser:

in	(inches, polegadas; 1pol = 25,4mm)
cm	(centímetros; 1cm = 10mm)
mm	(milímetros)
pt	(pontos; 1pt = 1/72pol)
px	(pixels)
em	(quadrado, letra M em tipografia; 1em = 16px)
%	(porcentagem)

Para exibição de documentos na tela, as melhores medidas são **px**, **em** e **%**. Para documentos impressos as medidas mais recomendadas são **em**, **cm**, **mm**, **in** e **%**.

17.5 - Tabela de Seletores

A seguir, uma tabela com alguns seletores, seus atributos, valores e exemplos:

font-size	Define o tamanho da fonte	
{ font-size: 12pt; }		
font-family	Seleciona uma família de fontes. Se uma falhar carrega a próxima.	nomes das fontes do windows, separadas por vírgulas
{ font-family: arial, verdana, script; }		
font-weight	Define a espessura da fonte (negrito).	normal; bold; 100,200,...900
{ font-weight: bold; }		
font-style	Aplica, ou não, o efeito itálico.	normal; italic;
{ font-style: italic; }		
font-variant	Ajusta para fonte pequena e em maiúscula.	small-caps; initial;
{ font-variant: small-caps; }		
color	Define a cor do texto.	red; #FF0000; rgb(255,0,0);
{ color: #FF0000; }		
text-decoration	Aplica, ou não, efeitos no texto.	none; underline; overline; line-through;
{ text-decoration: underline; }		
text-align	Define o alinhamento do texto	left; center; right; justify;
{ text-align: right; }		
text-indent	Define uma tabulação para o início do parágrafo.	
{ text-indent: 0.5in; }		
text-shadow	Define uma sombra para o texto, com 3 parâmetros: horizontal vertical e cor	
{ text-shadow: 2px 2px #ff0000; }		
text-transform	Define texto em letras minúsculas, maiúsculas ou capitulada.	lowercase; uppercase; capitalize;
{ text-transform: uppercase; }		
letter-spacing	Define o espaçamento entre letras no elemento <H1> e <H2>.	
{ letter-spacing: 2px; }		
line-height	Define a distância entre as linhas de um parágrafo.	
{ line-height: 24pt; }		
margin-left	Define a margem esquerda da página.	
{ margin-left: 2in; }		

margin-right	Define a margem direita da página.		
{ margin-right: 1cm; }			
margin-top	Define a margem superior da página.		
{ margin-top: 20px; }			
margin-bottom	Define a margem inferior da página.		
{ margin-bottom: 20px; }			
background-color	Define a cor do fundo.	red; #FF0000; rgb(255,0,0);	
{ background-color: #FFFF00; }			
background-image	Define uma imagem para o fundo,		
{ background-image: url("imagem.png"); }			
background-repeat	Define a repetição, ou não, de imagem do fundo.	repeat; repeat-x; repeat-y; no-repeat	
{ background-repeat: repeat-x; }			
background-attachment	Define se uma imagem do fundo será fixa.	fixed; scroll; local;	
{ background-attachment: fixed; }			
background-position	Define a posição de imagem do fundo.	%X %Y; pxX pxY; left top;	
{ background-position: center bottom; }			
border-color	Especifica a cor da borda. Pode-se combinar vários.	red; #FF0000; rgb(255,0,0);	
{ border-color: red black yellow green; }			
border-style	Especifica o estilo da borda. Pode-se combinar vários.	dotted; solid; none; ridge; outset;	dashed; double; grove; inset;
{ border-style: dotted solid; }			
border-width	Especifica a espessura da borda. Pode-se combinar várias medidas em pixels.	thin; thick; medium; px;	
{ border-width: 1px 20px; }			

[Editar o EXERCÍCIO 15 do Caderno de Exercícios](#)

[Editar o EXERCÍCIO 16 do Caderno de Exercícios](#)

18 - Introdução à Responsividade

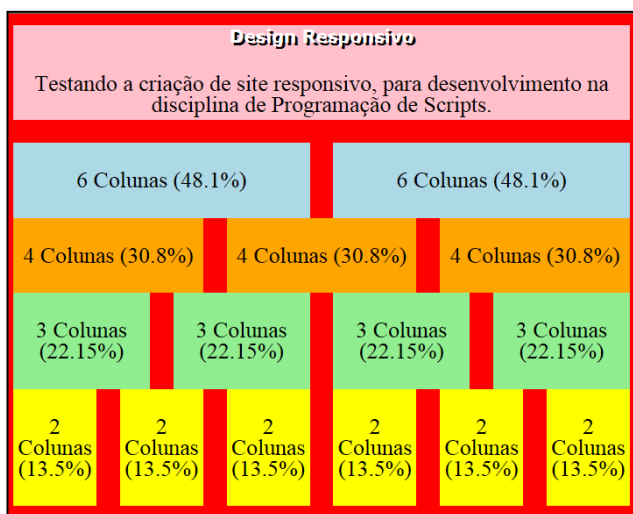
As páginas *web* podem ser visualizadas a partir de diferentes dispositivos como *desktops*, *tablets* e *smartphones*. Deve-se garantir que a página *web* seja corretamente visualizada e facilmente navegada, independente do dispositivo escolhido. As páginas *web* não devem deixar de exibir informações ou descartar algumas partes para se adequarem a dispositivos menores, mas sim adaptarem seu conteúdo para qualquer dispositivo. Esta é a tarefa do design responsivo, tornar a página *web* preparada para ser exibida nos diferentes tamanhos de tela.

É chamado de design web responsivo quando você usa CSS e HTML para redimensionar, ocultar, encolher, ampliar ou mover o conteúdo para que ele pareça bom em qualquer tela. Portanto, vamos utilizar aqui apenas HTML e CSS na construção de um modelo web responsivo que, apesar de não ser a solução absoluta para tudo é um bom caminho a ser seguido.

18.1 - Modelo Responsivo

Normalmente, web designers criavam a divisão principal com largura fixa (com 1140 pixels, por exemplo) que funcionava como um container organizando o conteúdo do *site* e impedindo que informações e objetos ficassem espalhados desordenadamente pela tela. Com o design responsivo jamais deve-se utilizar tamanhos ou posições fixas, sendo este o grande desafio.

A imagem abaixo mostra um design responsivo onde cada linha é subdividida em colunas e espaços entre as colunas. A seguir este modelo será descrito.



A *viewport* é a área visível do usuário em uma página *web*, e varia com o dispositivo, sendo menor em um *smartphone* do que em um *desktop*. Assim, define-se um elemento `<META NAME="viewport">` que captura instruções do navegador sobre como controlar as dimensões e a escala da página.

```
<HEAD>
  <META name="viewport"
    content = "width = device-width,
    initial-scale = 1.0">
</META>
```

O parâmetro `width = device-width` define que a largura da página *web* irá seguir a largura da tela do dispositivo, enquanto que `initial-scale = 1.0` define o nível de *zoom* inicial quando a página é carregada pela primeira vez no navegador.

Como a largura da tela varia em diferentes dispositivos, deve-se utilizar sempre tamanhos variáveis, medidos em porcentagens, por exemplo. Neste modelo, deve-se criar uma classe chamada `.container` que será nossa *viewport* e outra classe chamada `.linha` que agrupará as sub-divisões em colunas.

```
<STYLE>
.container {
  width: 100%;
  margin: 0 auto;
}
.linha {
  width: 100%;
  background-color: pink;
}
.fim {
  margin-right: 0%;
}
</STYLE>
```

Assim, deve-se pensar em uma *viewport* subdividida em colunas, onde cada coluna ocupa um percentual da largura da tela. Mas não deve-se esquecer que há espaço entre as colunas.

A seguir, o modelo com 12 colunas, onde cada coluna ocupa 4,85% da tela e possui um espaço de 3,8% da tela. Evidentemente que na última divisão à direita não haverá um espaço à direita, portanto foi criada uma classe chamada `.fim` que indica margem 0% à direita.



A partir deste modelo acima pode-se criar classes em CSS para construir agrupamentos de colunas que irão organizar as informações da página *web*, conforme segue:

a. Classe chamada `.cols2` com largura equivalente a 2 colunas de 4,85% + espaço de 3,8%, somando 13,5%:

```
<STYLE>
.cols2 {
  width: 13.5%;
  background-color: yellow;
  margin-right: 3.8%;
  float: left;
}
</STYLE>
```

Assim, pode-se criar uma estrutura HTML com 6 divisões lado a lado:

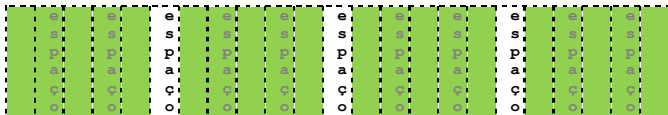


```
<div class="linha">
  <div class="cols2">
    <p>2 Colunas (13.5%)</p></div>
  <div class="cols2">
    <p>2 Colunas (13.5%)</p></div>
  <div class="cols2">
    <p>2 Colunas (13.5%)</p></div>
  <div class="cols2">
    <p>2 Colunas (13.5%)</p></div>
  <div class="cols2">
    <p>2 Colunas (13.5%)</p></div>
  <div class="cols2 fim">
    <p>2 Colunas (13.5%)</p></div>
</div>
```

- b. Classe chamada `.cols3` com largura equivalente a 3 colunas de **4,85%** + 2 espaços de **3,8%**, somando **22,15%**.

```
<STYLE>
.cols3 {
    width:          22.15%;
    background-color: lightgreen;
    margin-right:   3.8%;
    float:          left;
}
</STYLE>
```

Assim pode-se criar uma estrutura com 4 divisões conforme abaixo:

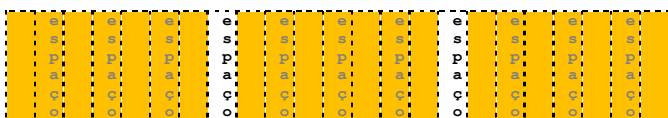


```
<div class="linha">
    <div class="cols3">
        <p>3 Colunas (22.15%)</p></div>
    <div class="cols3">
        <p>3 Colunas (22.15%)</p></div>
    <div class="cols3">
        <p>3 Colunas (22.15%)</p></div>
    <div class="cols3">
        <p>3 Colunas (22.15%)</p></div>
</div>
```

- c. Classe chamada `.cols4` com largura equivalente a 4 colunas de **4,85%** + 3 espaços de **3,8%**, somando **48,1%**.

```
<STYLE>
.cols4 {
    width:          30.8%;
    background-color: orange;
    margin-right:   3.8%;
    float:          left;
}
</STYLE>
```

Assim pode-se criar uma estrutura com 3 divisões conforme abaixo:

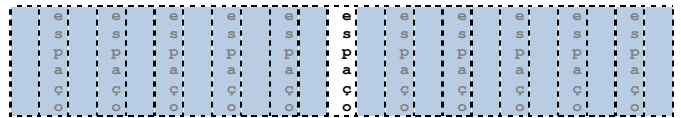


```
<div class="linha">
    <div class="cols4">
        <p>4 Colunas (30.8%)</p></div>
    <div class="cols4">
        <p>4 Colunas (30.8%)</p></div>
    <div class="cols4">
        <p>4 Colunas (30.8%)</p></div>
</div>
```

- d. Classe chamada `.cols6` com largura equivalente a 6 colunas de **4,85%** + 5 espaços de **3,8%**, somando **48,1%**.

```
<STYLE>
.cols6 {
    width:          48.1%;
    background-color: lightblue;
    margin-right:   3.8%;
    float:          left;
}
</STYLE>
```

Assim pode-se criar uma estrutura com 2 divisões conforme abaixo:



18.2 - Fontes em Modelo Responsivo

É desejável que o tamanho das fontes de uma página *web* com design responsivo também tenha tamanho variável, pois deve adequar-se aos diferentes tamanhos de dispositivos. Uma boa alternativa é utilizar a unidade de medida **em**, que representa o tamanho da letra **M** com origem nas fontes tipográficas de metal para impressão em papel, que eram utilizadas em tempos remotos. Esta é uma unidade de medida relativa, que varia proporcionalmente com o contexto, podendo ser utilizada para o tamanho das fontes e espaçamentos da página *web*. Pode-se considerar como padrão dos navegadores que:

1 em = 16 pixels

Sendo 16 pixels 100% do tamanho da fonte padrão, então tem-se que 10 pixels = 62,5%. Com esta definição, pode-se facilmente alterar o tamanho das fontes que foram escritas em **pixels** para **em**, por exemplo: **24px = 2.4em**. Para tanto, define-se em CSS:

```
<STYLE>
body {
    font-size: 62.5%;
}
</STYLE>
```

18.3 - Imagens em Modelo Responsivo

Também é desejável que o tamanho das imagens de uma página *web* com design responsivo tenha tamanho variável, se ajustando à variação da largura nos diferentes tamanhos de dispositivos. Isto deve se aplicar aos *imagens*, *backgrounds* e *vídeos*:

- a. **Imagens:** basta declarar os parâmetros ****. Mas se você já declarou todas imagens com tamanhos em pixels, pode-se utilizar o CSS para fazer a responsividade:

```
<STYLE>
img {
    width: 100%;
    height: auto;
}
</STYLE>
```

Também pode-se garantir que uma imagem nunca fique maior do que seu tamanho original, definindo tamanho 100% como largura máxima, com **max-width**:

```
<STYLE>
img {
    max-width: 100%;
    height: auto;
}
</STYLE>
```

b. Background: também pode-se garantir uma imagem de *background* responsiva utilizando o atributo CSS `background-size: (contain, cover ou x% y%);`:

- **contain** a imagem de fundo será redimensionada e tentará ajustar-se ao conteúdo, mantendo sua proporção original.
- **cover** a imagem de fundo será redimensionada e cortará partes para ajustar-se.
- **contain** a imagem de fundo será esticada para cobrir toda a área do conteúdo.

```
<STYLE>
div {
  width: 100%;
  height: 300px;
  background-image: url('img_normal.jpg');
  background-repeat: no-repeat;
  background-size: cover;
}
</STYLE>
```

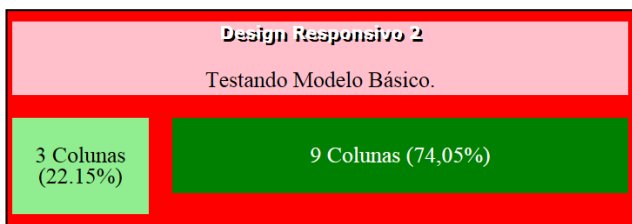
c. Vídeos: também pode-se garantir que uma área de exibição de vídeo seja responsiva:

```
<STYLE>
video {
  max-width: 100%;
  height: auto;
}
</STYLE>
```

```
<BODY>
<VIDEO width="500" controls>
  <source src="video_normal.mp4"
    type="video/mp4">
</VIDEO>
</BODY>
```

18.4 - Modelo Básico de *Grid* Responsivo

Pode-se criar diferentes modelos a partir da subdivisão em 12 colunas, como apresentamos anteriormente. Para tanto basta criar configurações de colunas cuja soma seja 12, e acrescentar a classe `.fim` na última coluna da direita. Como exemplo, vamos criar uma linha com uma divisão do lado esquerdo ocupando 3 colunas e uma divisão do lado direito ocupando 9 colunas.



Para tanto, cria-se uma classe chamada `.cols9` com largura equivalente a 9 colunas de **4,85% + 8 espaços de 3,8%**, somando **74,05%**:

```
<STYLE>
.cols9 {
  width: 74,05%;
  background-color: green;
  margin-right: 3.8%;
  float: left;
}
</STYLE>
```

```
<BODY>
<div class="container">
  <div class="linha">
    <H1>Design Responsivo 2</H1>
    <P>Testando Modelo Básico.</P>
  </div>
  <div class="linha">
    <div class="cols3">
      <p>3 Colunas (22.15%)</p></div>
    <div class="cols9 fim">
      <p style="color:white">
        9 Colunas (74,05%)</p></div>
    </div>
  </div>
</BODY>
```

18.5 - *Media Query*

A consulta de mídia (*Media Query*) é uma técnica de CSS que usa a regra `@media` para incluir um bloco de propriedades CSS, mas somente se uma condição for verdadeira. Por exemplo, a cor do fundo da página é definida como `lightgreen`, mas se a janela do navegador for menor que **600px** (*smartphones*), o fundo mudará para `lightblue`:

```
<STYLE>
body {
  background-color: lightgreen;
}
@media only screen and (max-width: 600px)
{
  body {
    background-color: lightblue;
  }
}
</STYLE>
```

Pode-se carregar imagens de *background* diferentes, de acordo com a largura da janela do navegador. Por exemplo, pode-se carregar uma imagem chamada `img_normal.jpg`, mas, se a janela do navegador ficar menor do que 400 pixels, então a imagem é substituída por `img_pequena.jpg`:

```
<STYLE>
body {
  background-image: url('img_pequena.jpg');
}
@media only screen and (min-width: 400px) {
  body {
    background-image: url('img_normal.jpg');
  }
}
</STYLE>
```

Pode-se adicionar um ponto de interrupção na remodelagem da tela. Isso é útil para tela de *smartphones* que são bem pequenas em relação aos monitores de *desktop* e devem ter um limite para o esmagamento das divisões exibidas na página. Por exemplo, se a janela do navegador for menor do que **768px** (*tablets*), haverá um ponto de interrupção. No exemplo abaixo, o parâmetro `width: 100%` será aplicado em todas as classes cujos nomes se iniciam com `cols`.

```
<STYLE>
@media only screen and (max-width: 768px) {
  [class*="cols"] {
    width: 100%;
  }
}
</STYLE>
```

Atualmente há uma grande demanda por navegação em dispositivos móveis como *smartphones* e *tablets* por exemplo, com largura de tela de até 768 pixels. Esta nova realidade pode levar o desenvolvedor *web* a configurar as páginas pensando primeiro em dispositivos móveis, criando design responsivo para janelas menores do que 768 pixels. Se a janela do navegador ficar maior, então a página se ajustará. Caso o desenvolvedor aplique este conceito, chamado de *Mobile First*, então troca-se **max-width** por **min-width**:

```
<STYLE>
@media only screen and (min-width: 768px) {
  [class*="cols"] {
    width: 100%;
  }
}
</STYLE>
```

O modelo responsivo apresentado na seção 16.1 teve como base uma largura de janela de 1440 pixels, com 12 divisões. Para construir um design responsivo *Mobile First* é aconselhável ter como base uma largura de 600 pixels, típica das telas dos smartphones, com 9 subdivisões, por exemplo. Mas esta é uma decisão do desenvolvedor.

Pode-se carregar diferentes folhas de estilo CSS - em arquivos externos - utilizando o parâmetro **media** na tag **<LINK>**, conforme segue:

**<LINK rel="stylesheet"
href="arquivo.css" type="text/css"
media="condição">**

A especificação da condição é feita pelos operadores lógicos de **@media** que são **and**, **not**, **or** e **only**. Assim, pode-se especificar diferentes folhas de estilos para diferentes tamanhos de janela do navegador:

```
<HEAD>
<!-- para desktop -->
<LINK href="estilo1.css" type="text/css"
rel="stylesheet"
media="screen and (min-width: 961px)" />

<!-- para tablet -->
<LINK href="estilo2.css" type="text/css"
rel="stylesheet"
media="screen and (max-width: 960px) and
(min-width: 481px)" />

<!-- para smartphone -->
<LINK href="estilo3.css" type="text/css"
rel="stylesheet"
media="screen and (max-width: 480px)" />
</HEAD>
```

Também pode-se alterar o *layout* de uma página de acordo com a orientação do navegador, podendo ser definido como **Portrait** (retrato) ou **Landscape** (paisagem).

```
<STYLE>
body {
  background-color: lightgreen;
}
@media only screen and (orientation: landscape) {
  body {
    background-color: lightblue;
  }
}
</STYLE>
```

Referências

<https://en.wikipedia.org>
<https://www.w3schools.com>
<http://www.tutorialspark.com>
<http://blog.popupdesign.com.br>
<http://www.devmedia.com.br>