

PROJETO E-JOVEM



EDUCANDUS

APOSTILA HTML-CSS-JS

Índice

1.	Introdução ao HTML	4
2.	Estrutura de um arquivo HTML	6
3.	Textos	10
4.	Recursos Visuais	13
5.	Frames	20
6.	DIV	23
7.	Formulários	25
8.	Desafio	30
9.	O que é CSS e como funciona.	31
10.	O JavaScript	51
11.	EXERCÍCIOS RESOLVIDOS	72
12.	Exercício Proposto HTML – CSS -JS	98

Introdução

Nesta apostila vamos abordar a Linguagem básica de internet HTML, mas antes de começarmos os estudos do html vamos falar um pouco sobre o famoso **www**.

www: Também conhecido como Web é a interface gráfica da grande rede de comunicação à internet. Sua idéia é criar um conjunto de informações sem fronteiras, tendo as seguintes características:

- Interface consistente;
- Incorporar um vasto conjunto de tecnologias e tipos de documentos;
- E ter uma leitura universal;

E para isso faz-se necessário as seguintes ferramentas:

- HTTP: protocolo de transmissão de dados;
- URL: sistema de endereçamento;
- HTML: linguagem de formatação para transmitir documentos através da rede;

1. Introdução ao HTML

1.1. O que é HTML?

HTML – Hyper Text Markup Language (Linguagem de marcação de Hypertexto.) é uma linguagem considerada a base de todas as outras linguagens de desenvolvimento de projetos para Web que trabalha com marcação de textos e é muito utilizada atualmente para a criação de páginas Web.

Com ela você pode compartilhar fotos, vídeos, músicas, textos e fazer muitas coisas.

Documentos HTML podem ser interpretados por navegadores como: internet explorer, firefox, safari, opera entre outros.

1.2. História

Foi a primeira linguagem no setor e é usada amplamente até os dias atuais. Foi criada pelo Tim Berners-Lee.

Html surgiu da união dos padrões Hytime e SGML

HyTime: é um padrão de representação estruturada de hipermídia e conteúdo baseado em tempo (áudio, vídeo etc.). É também um padrão independente de processamento de textos em geral.

SGML: não foi desenvolvido para hipertexto, mas tornou-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações.

Existe muita coisa relacionada com a história do HTML. Mas eu nesta apostila pretendo apenas ensinar como usá-lo.

1.3. Tags

Uma página básica em HTML e um arquivo de texto simples contendo o texto da página e os comandos HTML que definem a formatação das mesmas. Os comandos são indicados entre os marcadores chamados TAG que vem entre os símbolos de < e >.

A maioria das tags tem uma tag correspondente para o fechamento:

```
<tag> ... </tag>
```

Uma tag é formada, por comandos atributos e valores. Os atributos modificam os resultados padrões dos comandos e os valores definem estas mudanças.

```
<HR color="red">
```

Importante: HTML não faz diferença entre maiúsculas e minúsculas (não é "case sensitive"). Então a notação <title> é equivalente a <TITLE> ou <TiTlE>.

1.4. Editores HTML

Existem muitos editores que podem ser utilizados para editar um arquivo HTML

- Quanta plus
- Adobe DreamWeaver
- NVU
- Vim

1.5. W3C

Consortio de empresas de tecnologia, formado por 500 membros. Desenvolve padrões para criação e interpretação de conteúdos para web. Sites desenvolvidos utilizando o padrão w3c podem ser acessados e visualizados por quaisquer pessoas, independente do dispositivo utilizado.

1.6. Validar HTML no W3C

O site do w3c contém a descrição de todos os padrões por ele definido. <http://www.w3c.org>. É possível validar um html que esteja na internet, para saber se ele esta ou não no padrão w3c.

Para isso basta acessar o site <http://validator.w3.org>, basta colocar a url preenchendo o campo address com a url de um site qualquer e clicar no botão check.

1.7. Extensões HTML

Com o passar dos anos novos padrões são baseados em html foram surgindo, segue abaixo dois novos padrões:

1.7.1. DHTML: Dynamic HTML é a união de Html, JavaScript e CSS aliada a um DOM (Document object Model – modelo de objetos de documentos) para permitir que a página web seja modificada dinamicamente na máquina do cliente sem que seja necessário fazer uma nova solicitação ao servidor.

1.7.2. XHTML: é uma reformulação da linguagem de programação HTML baseada em XML que combina as tags de marcação HTML com as regras XML. Este processo de padronização tem em vista a exibição de páginas web em diversos dispositivos (celular, palm, celular) com intenção de melhorar a acessibilidade.

2. Estrutura de um arquivo HTML

2.1. Macro Estrutura

Abaixo mostramos a estrutura básica de um arquivo HTML:

```
<html>
  <head>
    <title>Título da página</title>
  </head>
  <body>
  </body>
</html>
```

2.2. Descrição

- **Tag <HTML>**: Define o início do documento, informando ao navegador que a partir daquele ponto ele deve interpretar como um documento HTML.
- **Tag <HEAD>**: Define o cabeçalho do documento onde estão contidas todas as informações sobre o documento que está sendo lido ou aberto.
- **Tag <BODY>**: Define o início do corpo do documento, tudo que será apresentado pelo navegador estará nessa tag.

2.3. Cabeçalho - <HEAD>

Agora veremos as principais tags que podemos encontrar dentro do cabeçalho.

- A principal é a **tag <title>...</title>**, que como no nome já diz, define o título do documento este é o texto que aparece na barra superior do navegador.

Exemplo:



Sempre escolha títulos coesos para suas páginas html, pois Quando o usuário entra numa ferramenta de busca como Google, Yahoo, Cade, etc procurando por um assunto específico, o conteúdo da tag <TITLE>.

Dica: Coloque títulos com ponto na frente assim quando o usuário adicionar a sua página aos Favoritos e caso este esteja em ordem alfabética o seu Título ficara nas primeiras posições da lista. E terá mais chance de ser acessado.

Exemplo: ...: Título da Página ::... ou ::... Título da Página ...:

- Tag <meta>...</meta>, mas conhecidas como meta tags não são obrigatórias, mas algumas ajudam os mecanismos de busca a encontrar seu site com mais facilidade.

As tags META tem dois atributos principais:

NAME: Esta indica um nome para informação

HTTP-EQUIV: este pode ser utilizado para que seja feita uma configuração e/ou alteração no comportamento de navegador.

Para o atributo http-equiv, poucos valores são utilizados, o mais comum é o content-type que informa ao navegador o conjunto de caracteres que o documento está usando fazendo com que o navegador apresente facilmente os caracteres corretos.

Exemplo:

```
<head>
  <title>Aula Html</title>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <meta name="Description" content="Exemplo desenvolvido na sala de aula">
  <meta name="KeyWords" content="HTML aula Cabeçalho">
</head>
```

Ainda no cabeçalho podemos fazer referência a algum arquivo externo que precisaremos utilizar durante a execução do nosso documento. Veremos com mais detalhes nas aulas de CSS e JavaScript.

Exemplo:

```
<head>
  <title>Aula Html</title>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
  <meta name="Description" content="Exemplo desenvolvido na sala de aula">
  <meta name="KeyWords" content="HTML aula Cabeçalho">
  <link rel="stylesheet" href="folha_estilo.css">
  <script type="text/javascript" src="arquivo_de_script.js"></script>
</head>
```

2.4. Corpo - <BODY>

Determinados pelos marcadores <BODY>...</BODY>, o corpo e a parte da página que contém informações que serão visualizadas na tela. Assim como as outras tags ela contém alguns atributos, que mostramos abaixo:

- **BGCOLOR:** Determina a cor do Fundo da página, a cor padrão é branco ou cinza.
- **Text:** Determina a cor do texto da página. A cor padrão é preta;
- **Link:** Determina a cor dos links na página. A cor padrão é azul;
- **Alink:** Indica a cor dos links ao serem acionados. A cor padrão é vermelha.
- **Vlink:** Indica a cor dos links depois de visitados. A cor padrão é Azul escuro ou Roxo.

```
<BODY BGCOLOR="#FFFFFF" TEXT="#000000" LINK="#000000" ALINK="#000000"
VLINK="#000000" BACKGROUND="URL">
```

Agora conheceremos rapidamente algumas tags utilizadas para exibir conteúdos no navegador:

- `<h1><h2>...<h6>`: Cabeçalho e títulos a serem apresentados no documento.
- `
`: quebra de linha
- ``: Formata um texto (cor, tamanho, fonte).
- ``: negrito
- `<i>`: itálico
- `<u>`: sublinhado
- `<s>`: riscado
- `<table>`: cria uma Tabela onde linhas são com `<tr>` e com colunas com `<td>`
- `<div>`: Determina uma divisão na página e pode assumir várias formatações
- `<p>`: Cria um novo parágrafo
- ``: Adiciona uma imagem
- `<a>`: criação uma hiper ligação de uma lugar para outro
- `<input>`: campo de interação com o usuário
- `<cite>`: citação
- `<form>`: define um formulário
- `<adress>`: Endereço
- `<textarea>`: caixa de texto com linhas e colunas
- `<acronym>`: acrônimo (sigla).

2.5. Olá Mundo

Vamos agora por a mão na massa e criar nossa primeira página utilizando o que aprendemos até agora. É muito importante que você crie uma pasta no seu computador onde ficarão armazenados os arquivos que vamos criar durante todo o curso.

Siga os passos abaixo para criarmos nosso primeiro exemplo:

- Abrir um editor de textos simples, o que você se adaptar melhor.
- Criar um novo arquivo e salva-lo como `olaMundo.html`
- Escrever as tags básicas do html(tags que vimos anteriormente).
- Dentro da Tag `<body>` escrever as tags que serão apresentadas.
- Salvar o arquivo.
- Ir ate o navegador, pedir para abrir um arquivo e selecionar o arquivo criado.
- Conferir o resultado.

Tente ser o mais criativo possível e não faça apenas um simples “olá mundo”, teste as mais variadas possibilidades de combinação com o que aprendemos.

---- Modelo HTML ----

3. Textos

3.1. Inserção de textos

Qualquer texto escrito dentro do limites da tag `<body>...</body>` serão exibidos em tela, a menos que seja um comentário, que deve ser escrito da seguinte forma:

`<!-- Comentário -->`

```
<html>
<head>
  <title>Aula Html</title>
  <meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
  <meta name="Description" content="Exemplo desenvolvido na sala de
aula">
  <meta name="KeyWords" content="HTML aula Cabeçalho">
  <link rel="stylesheet" href="folha_estilo.css">
  <script type="text/javascript"
src="arquivo_de_script.js"></script>
</head>
<body >
  comentário 1
  <!--comentário 2-->

</body>
</html>
```

No exemplo acima apenas a expressão “comentário 1”, será exibida na tela, o que estiver comentado será ignorado pelo navegador.

3.2. Formatação de textos

Existem dois modos de formatar um texto HTML:

- **Formatação Lógica:** É aquela que segue o significado lógico do texto marcado, se estiver marcado como cabeçalho, ou um endereço, ou uma citação irá seguir o padrão determinado por essas tag, lembrando que podemos alterar essas configurações, a apresentação vai variar de acordo com o navegador e suas configurações.
- **Formatação Física:** É aquela em que o programado define exatamente como deve ser apresentada, definindo tamanho, cor, fonte, alinhamento, etc. Não podendo ser alterada pelo usuário final.

Formatação Básica:

Cores: Para alterar cores vamos usar a tag ` ` com um atributo chamado **color**, lembrando que as cores são definidas pelo padrão RGB.

`Este texto vai aparecer em vermelho`

Fontes: Para alterar a fonte utilize a mesma tag `` agora utilizando o atributo **face**, lembrando que a frase só será apresentada na fonte escolhida se o computador no qual o documento html esteja sendo executado, tenha a fonte instalada.

`Este texto vai aparecer em vermelho`

Tamanho: Para alterar o tamanho do texto utilizamos a mesma tag `` agora utilizaremos a atributo **size**.

`texto vermelho`

`texto vermelho`

3.3. Blocos

Até agora vimos a maneira básica de se formatar um texto, vou mostrar a você agora uma forma de editar textos em bloco.

- **`<PRE></PRE>`:** Esta tag apresenta o texto da forma como ele foi apresentado.

Ex: `<pre>texto`

digitado

totalmente

solto `</pre>`

Cuidado: pois o texto pode ultrapassar os limites do navegador de quem estiver visualizando o arquivo html

- **`<ADDRESS></ADDRESS>`:** A tag `<adress>` é utilizado para indicar o endereço (postal, e-mail ou ambos) do autor do documento HTML.

Ex: e-mail`<address>`educandus@educandus.com.br`</adress>`

Uma dica é que sempre que apresentarmos uma tag nova você insira no arquivo html de teste que você criou para realizar os testes.

3.4. Cabeçalhos

Existem 6 níveis de cabeçalho. Numerados de H1 a H6, que são exibidos em fonte maior que a fonte normal. Os marcadores de titulo podem ser alinhados através do atributo align, veja alguns exemplos abaixo:

```
<html>
<head>
  <title> Títulos </title>
</head>
<body>
  <H1 align='center'> Aqui vai o título 1</H1>
  <H2 align='left'> Aqui vai o título 2</H2>
  <H3 align='right'> Aqui vai o título 3</H3>
  <H4> Aqui vai o título 4</H4>
  <H5> Aqui vai o título 5</H5>
  <H6> Aqui vai o título 6</H6></body>
</html>
```

A prioridade dos mecanismos de busca é do menor para o maior.

3.5. Seperadores

Agora veremos alguns separadores que nos ajudam a organizar os nosso textos:

- `
`: Este seperador serve para quebrar uma linha em determinado ponto, pois os navegadores já quebram linhas quando apresentam textos.

Ex: Este é um texto quebrado aqui

E continua aqui.

- `<hr>`: Esta tag desenha uma linha horizontal e possui alguns atributos que podemos personalizar. O padrão para a linha é de uma linha sombreada.

Ex: `<HR WIDTH="200" SIZE="4">`
`<HR WIDTH="50%" ALIGN="left">`
`<HR SIZE="8" COLOR="#00FF00" ALIGN="right" NOSHADE>`

- `<p>`: tag que serve para marcar um parágrafo, pode ser utilizado com a tag `
` e possui atributos de alinhamento igual ao cabeçalho.

Ex: `<p align='center'>textos textos</p>`
`<p align='right'>textos textos textos</p>`

3.6. Outras tags de formatação

- `<i>`: Deixa um texto em itálico.

Ex: Essa<i>palavra</i>está em itálico

- `<u>`: Deixa a palavra ou texto sublinhado

Ex: Essa<u>palavra</u>está sublinhado

- `<s>`: Deixa a palavra ou texto riscado

Ex: Essa<s>palavra</s>está riscada

- : Deixa a palavra em negrito

Ex: Essa< b >palavra</ b >está em negrito

3.7. Link

Como nos já sabemos existem nas mais diversas páginas da web inúmeros links para outras páginas, serviços e downloads. O conceito é simples: Proporcionar a opção de sair do documento atual para um outro documento definido no link.

Para criarmos um link utilizamos a tag <a> e um atributo chamado href, que vai conter o caminho do arquivo do destino, ou seja, o arquivo para qual nós vamos ao clicar neste link.

Abaixo o exemplo de como fazer:

Ex:

Para ir para o site da educandus, basta clicar

`Aqui`

4. Recursos Visuais

4.1. Imagens

Na internet é comum encontrarmos imagens nas páginas e agora veremos como fazer isso em nossa página. A imagem a ser inserida na página é um arquivo que preferencialmente deve estar no formato GIF ou JPG, mas isso não quer dizer que você não possa colocar outros formatos como PNG, BMP, e outros.

Mas não devemos apenas escolher o nome mais bonito e pronto, cada um destes tipos de imagens tem suas particularidades e existe o momento mais adequado para usar cada tipo. Veja abaixo as particularidade dos dois principais tipos de imagens.

- **JPEG:** Devemos utilizar jpeg para imagens complexas e com muitas variações de cores como fotos, imagens muito coloridas, imagens em tons de cinza. Sempre lembrar que em imagens complexas devemos utilizar JPEG.
- **GIF:** Devemos utilizar gif quando tivermos imagens com cores sólidas e poucas variações como imagens em preto e branco (sem tons de cinza), imagens com cores lisas (sem nuances), neste tipo de imagem o gif consegue um melhor resultado visual e uma melhor compactação.

4.2. Inserção de Imagens

Para se colocar imagens em uma página HTML. Usa-se a tag , que tem como principal atributo o SRC que significa SOURCE, ou seja, o caminho da imagem que queremos exibir.

Ex: Abaixo você verá uma foto de um moderno notebook:
.

Vimos no exemplo que adicionamos um atributo ALT que deve conter uma breve descrição da imagem a ser apresentada, pois caso ela não venha a aparecer o usuário vai receber esta descrição e não ficará apenas o espaço vazio e também servirá como um hint que ao passar o mouse em cima exibe o conteúdo do alt.

4.3. Lista de definição

Em html temos o recurso de criar listas como em alguns editores de texto, estas podem ser apenas marcadas, numeradas, tabuladas entre outras.

Listas de Definição

```
<DL>
  <DT> Linha 1
  <DT> Linha 2
  <DD> Linha 2.2
  <DD> Linha 2.3
  <DT> Linha 3
  <DD> Linha 3.1
</DL>
```

Esta lista vai apenas tabular o conteúdo das linhas onde os DT ficam mais para a esquerda que os DD.

Lembre-se sempre de aplicar os novos conhecimentos ao arquivo de exemplo que você mesmo criou.

4.4. Lista Marcada ou Não-numeradas

Esta lista é uma das mais comuns ela é marcada por bolinhas no início da linha. Para alterar os marcados use o atributo type que pode ser dos seguintes tipos: "DISC" , "CIRCLE" ou "SQUARE".

Obs.: Caso não especifique o TYPE do UL será colocado automaticamente o símbolo "DISC" como aconteceu no exemplo acima. O "DISC" será uma bolinha preta, o "CIRCLE" será uma bolinha sem preenchimento e o "SQUARE" será um quadrado preto.

Exemplo 1:

```
<ul>
  <li>Linha 1</li>
  <li>Linha 2</li>
<ul>
  <li>Linha 2.2
  <li>Linha 2.3
</ul>
  <li>Linha 3</li>
  <li>Linha 4</li>
</ul>
```

Exemplo 2:

```
<ul type="circle">
  <li> Curitiba </li>
  <li> São Paulo </li>
  <li> Rio de Janeiro </li>
</ul>
```

Exemplo 3:

```
<ul type="square">
  <li> Curitiba </li>
  <li> São Paulo </li>
  <li> Rio de Janeiro </li>
</ul>
```

Adicione isso ao arquivo teste e veja o resultado.

4.5. Listas numeradas

Agora vamos criar uma lista numerada,

Exemplo 1:

```
<OL>
  <LI> Linha 1
  <LI> Linha 2
  <OL>
    <LI> Linha 2.1
  </OL>
```

```
<LI> Linha 3
</OL>
```

Executando seu arquivo html você vai notar que a lista numerada não gera índices como 2.1 que era esperado em nossa lista acima. Mas para isso utilizamos o atributo TYPE com um dos valores: a, i ou I que ficaria assim:

Exemplo 2:

```
<OL>
<LI> Linha 1
<LI> Linha 2
  <OL TYPE=I>
    <LI> Linha 2.1
    <LI> Linha 2.2
  </OL>
<LI> Linha 3
</OL>
```

4.6. Sub-Listas

Já sabemos que podemos aninhar listas, assim criando sub-listas, nos exemplos anteriores sempre utilizamos listas iguais, mas podemos aninhar uma lista numerada com uma lista marcada ou com uma lista não-numerada. Veja os exemplos:

Exemplo 1:

```
<OL>
<LI> Linha 1
<LI> Linha 2
  <UL TYPE=circle>
    <LI> Linha 2.1
    <LI> Linha 2.2
  </UL>
<LI> Linha 3
</OL>
```

Neste temos uma lista Numerada com uma lista marcada.

Exemplo 2:

```
<DL>
<DD> Linha 1
<DD> Linha 2
  <UL TYPE=circle>
    <LI> Linha 2.1
    <LI> Linha 2.2
  </UL>
<DD> Linha 3
</DL>
```

Agora temos uma lista tabulada com uma lista marcada.

4.7. Tabelas

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.

É utilizada da mesma forma que nos editores de texto como o Writer, Word. Tabelas contêm linhas e colunas todas representadas por TAGs dentro de uma TAG que define a tabela.

Uma tabela é delimitada com as tags `<TABLE></TABLE>`, sendo que dentro destes marcadores são colocadas as linhas e as colunas da tabela, veja o exemplo:

Exemplo 1:

```
<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>
```

Exemplo 2 :

Tabelas com outras tags.

```
<TABLE border=1> ← Está tag indica o início da tabela.
  <TR> ← Está defini o início de uma linha.
    <TD><b>Coluna 1</b></TD> ← Está defini uma Coluna.
    <TD><b>Coluna 2</b></TD>
  </TR>
  <TR>
    <TD>Valor 1 Linha 1</TD>
    <TD>Valor 2 Linha 1</TD>
  </TR>
  <TR>
    <TD>Valor 1 Linha 2</TD>
    <TD>Valor 2 Linha 2</TD>
  </TR>
</TABLE>
```

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

4.7.1. Principais Atributos de uma tabela

- **Border:** 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.
- **Bordercolor:** Como o nome diz cor da borda que é definida no padrão RGB que vimos em aulas anteriores.
- **Height:** Este define a altura da tabela em pixels ou em porcentagem.
- **Width:** Este define a largura da tabela em pixels ou em porcentagem.
- **Background:** Este permite colocar uma imagem como fundo da tabela, basta apenas adicionar como valor o link ou caminho da imagem desejada.
- **Bgcolor:** Este define apenas uma cor de fundo para a tabela.
- **Cellpadding:** Define o espaço entre a borda e o conteúdo da célula. Sempre em pixels.
- **Cellspacing:** Define o espaço entre as bordas das células. Sempre em pixels.

Estes são os principais! Agora teste todos eles, em nosso arquivo de exemplo.

4.7.2. Tabelas Aninhadas

Podemos utilizar uma tabela aninhada em outra e quantas vezes se fizer necessário.

Veremos agora um exemplo de tabelas aninhadas:

```
<TABLE border=1>
  <TR>
    <TD><b>Coluna 1</b></TD>
    <TD><b>Coluna 2</b></TD>
  </TR>
  <TR>
    <TD>Valor 1 Linha 1</TD>
    <TD>Valor 2 Linha 1</TD>
  </TR>
  <TR>
    <TD>Valor 1 Linha 2</TD>
    <TD>
      <table border=1>
        <tr>
          <td> Linha 1 tabela 2</td>
          <td>Valor 1 tabela 2</td>
        </tr>
        <tr>
          <td>Linha 2 tabela 2</td>
```

```
        <td>Valor 2 tabela 2</td>
    </tr>
</table>
</TD>
</TR>
</TABLE>
```

Usamos a tabela do primeiro exemplo e na última célula adicionamos uma outra tabela.

Agora faça o mesmo no seu arquivo exemplo.

4.7.3. Utilização das Tabelas

Antigamente utilizávamos tabelas para montar layouts. Foi por muitos anos a melhor forma de construir websites, mas atualmente dão lugar ao “tableless”.

Utilizamos tabelas atualmente para apresentar dados de forma organizada e ordenada.

Exemplo:

```
<html>
<head>
<title>Layout com HTML</title>
</head>

<body>
    <table width="600" cellpadding="0" cellspacing="0">
        <tr>
            <td><h1>Topo</h1></td>
        </tr>
        <tr>
            <td>
                <table height="300" width="100%" cellpadding="0" cellspacing="0">
                    <tr>
                        <td width="100" valign="top">MENU</td>
                        <td valign="top">Conteúdo</td>
                    </tr>
                </table>
            </td>
        </tr>
        <tr>
            <td align="center"><address>Rodapé</address></td>
        </tr>
    </table>
</body>
</html>
```

Este é um exemplo de como podemos montar layouts com tabelas, na maioria das vezes utilizamos imagens que encaixam de forma harmônica e tornam a beleza do site bem superior aos feitos apenas com html.

5. Frames

Os Frames são divisões da tela do browser. Com isso, torna-se possível apresentar mais de uma página por vez: por exemplo, um índice principal em uma parte pequena da tela, e os textos relacionados ao índice em outra parte.

Na aula passada falamos que atualmente utilizamos tableless para montar layouts e que antes se usava tabelas e antes das tabelas usávamos frames.

Mas eles continuam com sua serventia como, por exemplo, criar um HELP para um software que fique on-line, podemos dividir nossa tela em duas uma mais fina onde ficaria o índice e outra onde ficaria o conteúdo.

Nem todos os usuários gostam deles, pois nem sempre a navegação é fácil, além de problemas para a impressão e a marcação dos documentos interiores aos Frames nos Bookmarks.

Vamos ver agora como montaríamos um help.

Para isso precisaremos criar 3 arquivos dentro de uma mesma pasta. Então vamos começar, crie uma pasta chamada frames e dentro dela crie um arquivo chamado index.html

Este arquivo vai conter uma divisão da tela que é definida pela tag FRAMESET junto com os atributos cols ou rows, cols para definir colunas e rows para definir linhas.

Dentro da tag <frameset> teremos as tags <frame> que vão definir os arquivos que serão apresentados e em que posição. Um detalhe importante é todas as tags frame devem conter um atributo name contendo o nome deste frame para utilizarmos no futuro, e devemos colocar no atributo src o caminho do arquivo que será apresentado.

Exemplo parte 1: index.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Help do Software</title>
  </head>
  <frameset cols='20%, 80%'>
    <frame src='indice.html' name='indice'>
    <frame src='home.html' name='conteudo'>
    <noframe>
      <h1>Este navegador não suporta FRAMES!</h1>
```

```
        </noframe>
    </frameset>
</html>
```

Exemplo Parte 2: home.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Home do Help</title>
  </head>
  <body>
    <center>
      <h1>Bem Vindos!</h1>
      <h2>Help versão 1.0</h2>
    </center>
  </body>
</html>
```

Exemplo Parte 3: indice.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Home do Help</title>
  </head>
  <body>
    <ul>
      <li> Home
      <li> Assunto 1
      <li> Assunto 2
    </ul>
  </body>
</html>
```

Crie os três arquivos do jeito que foi colocado no exemplo e execute o index.html, veja o resultado.

5.1. Links com Frames

Para criar links vamos utilizar a tag `<a>` normalmente só precisaremos adicionar um atributo chamado **target** que vai conter o valor do nome do frame onde este arquivo será aberto.

Para isso vamos abrir o arquivo indice.html e criar os links dentro da lista que já existe.

```
<ul>
  <li> <a href='home.html' target='conteudo'>Home</a>
  <li> <a href='assunto1.html' target='conteudo'>Assunto 1</a>
  <li> <a href='assunto2.html' target='conteudo'>Assunto 2</a>
</ul>
```

Para que tudo funcione precisamos criar os arquivos assunto1.html e assunto2.html.

5.2. Frames Aninhados

Sim é possível aninhar frames veja abaixo alguns exemplos faça no seu arquivo teste e veja o resultado.

Frame com 2 Colunas:

```
<FRAMESET COLS="20%, 80%">
  <FRAME SRC="coluna_1.html">
  <FRAME SRC="coluna_2.html">
</FRAMESET>
```

Frame com 2 Linhas:

```
<FRAMESET ROWS="20%, 80%">
  <FRAME SRC="linha_1.html">
  <FRAME SRC="linha_2.html">
</FRAMESET>
```

Frame com 1 Linha e 2 colunas:

```
<FRAMESET ROWS="20%, 80%">
  <FRAME SRC="linha_1.html">
  <FRAMESET COLS="20%, 80%">
    <FRAME SRC="coluna_1.html">
    <FRAME SRC="coluna_2.html">
  </FRAMESET>
</FRAMESET>
```

Frame com 1 Coluna e 2 linhas:

```
<FRAMESET COLS="20%, 80%">
  <FRAME SRC="coluna_1.html">
  <FRAMESET ROWS="20%, 80%">
    <FRAME SRC="linha_1.html">
    <FRAME SRC="linha_2.html">
  </FRAMESET>
</FRAMESET>
```

5.3. Atributos

Vou agora mostrar mais dois atributos um da tag frame e outro da tag frameset são eles:

- **SCROLLING**: usado na tag frame e recebe como valor “yes” ou “no” que serve para ativar ou desativar a barra de rolagem neste frame.
- **FRAMEBORDER**: usado na tag frameset que também recebe o valor de “yes” ou “no” que apresenta ou não a borda dos frames.

Agora aplique estes dois atributos em nossos exemplos para ver o resultado.

6. DIV

Vamos agora falar de um dos recursos mais utilizados em páginas html, Muitos a consideram como uma integrante do grupo das tags malucas, que além da tag div tem as seguintes tags SPAN e Blockquote.

A tag "div" é utilizada para configuração de blocos de textos, principalmente quando se trata de CSS, mas normalmente atribuímos a ela formatações mais complexas.

Basicamente quando utilizamos o DIV ele cria uma quebra de linha antes e depois do grupo de elementos ali selecionado.

Agora crie uma nova pasta e crie um documento exepmlosDiv.html

Dentro deste html crie algumas DIVs como está apresentada abaixo:

```
<div>
  <h1>Nome do Meu Site</h1>
  <h2>Slogan do meu site</h2>
</div>
<div>
  Home | Clientes | Contato
</div>
<div>
  <h3>Bem Vindos</h3>
  <span><p>Este é um exemplo da utilização de <b>div</b> e
  <b>span</b>.</p></span>
</div>
```

Você deve ter percebido que usei a tag `` eis a explicação do que ele faz: Ela é Utilizada normalmente para agrupar elementos em uma linha. Uma quantidade menor.

6.1. Atributos

Vamos a alguns atributos que temos na div e no span:

- **Align:** Alinha horizontalmente a div.

Agora podemos colocar na div que representa nosso menu um `align = right` e veremos que nosso menu vai agora está do lado direito do navegador.

Ex: `<div align="right">`

- **Id:** Este atributo identifica uma tag, pode ser usado em várias tags não só na div, ele é único e seu valor não pode ser repetido.

Visualmente não tem nenhum efeito para o usuário final. Mas vai ser muito útil quando tivermos estudando CSS.

Ex: `<div id="menu" align="right">`

Agora vamos nomear nossas divs com os nomes de topo, menu e corpo.

- **Style:** Este é outro atributo que se utiliza bastante nas divs mas ele deve conter instruções de CSS dentro para que tenha efeito então utilizaremos este atributo depois.
- **Class:** É outro atributo destinado ao uso de CSS que também estudaremos depois.

6.2. Tableless

Tableless como sabemos é a forma mais atual de montar layouts de web sites, mas o que vem a ser tableless?

Tableless é uma metodologia de desenvolvimento que utiliza basicamente DIV + CSS.

As maiores vantagens em utilizar tableless é que o conteúdo da página será mostrado mais rápido, mas isso não significa que a página está mais leve, outra boa vantagem é a acessibilidade, seu site poderá ser visto em dispositivos móveis sem nenhuma alteração e também pode ser utilizado por deficientes visuais embora não sejam todos os sites que tem está característica.

Agora que sabemos utilizar DIVs vamos montar a parte html de um layout tableless que será completado quando aprendermos CSS.

Devemos montar um html que fique igual a este da imagem apresentada abaixo:

É importante que cada umas das partes do site estejam em uma div.

Exemplo:

```
<div id=geral> ← Aqui é uma div geral, onde vai está todo o site.  
  <div id=topo> ← Todo o topo do site.  
    <div id=marca> ← Apenas a logo marca  
    </div>  
    <div id=menu> ← Todo o menu do site.  
  </div>  
</div>  
</div>
```

Este foi um exemplo de como devem está estruturadas as DIVs.

7. Formulários

Formulários estão presentes na Internet para possibilitar cadastros, pesquisas, envio de comentários, aumentando o poder de interação com os visitantes dos sites. Um formulário HTML e uma pagina Web que contem, alem de texto, elementos especiais chamados controles, representados por caixas de checagem, botões, caixas de seleção, campos de textos, etc.

Um formulário funciona assim: Os usuários preenchem os campos do formulário submetendo-o, em seguida, a algum agente de processamento. Neste momento, todas as informações fornecidas são enviadas a um programa escrito especialmente para processar esses dados de acordo com alguma necessidade e especificação. Em alguns casos os dados são gravados em um Banco de Dados (BD), em outros casos uma nova pagina e construída, em outros ainda as informações são encaminhadas via e-mail.

A tag de formulários é a <FORM> que possui alguns atributos essenciais para seu funcionamento como, por exemplo:

- **METHOD:** Diz qual o método utilizado para enviar o conteúdo do formulário ao servidor, pode ser GET ou POST.
- **ACTION:** Diz para quem vamos enviar o conteúdo do formulário.

Vale salientar que a tag FORM sozinha não tem nenhum efeito visual para o usuário final, ela apenas agrupa todos os campos de INPUT para que sejam enviados juntos.

7.1. GET OU POST

Todos os dois métodos enviam os dados dos formulários para o servidor, mas com algumas diferenças.

No método **GET** os dados entrados no formulário fazem parte da URL (endereço), e suporta até 128 caracteres.

No método **POST** os dados são enviados no corpo da mensagem que vai para o servidor, isso possibilita o envio de uma grande quantidade de dados.

7.2. Input

O campo INPUT é um dos campos que utilizamos para atribuir valores aos nossos formulários ele tem um atributo chamado TYPE que aceita 9 valores diferentes, este atributo define o tipo de campo que será apresentado ao usuário, outro atributo importante é o NAME que identifica o campo para que possamos utiliza-lo depois.

O atributo TYPE pode conter um destes valores:

- **TEXT:** Um campo de texto simples.
- **PASSWORD:** Uma campo de senha, quando digitamos ele aparece “*”.
- **CHECKBOX :** Campo para múltipla escolha.
- **RADIO:** Campo de escolha única.
- **SUBMIT:** Botão que envia os dados para o servidor.
- **RESET:** Volta todos campos do formulário ao seu valor inicial.
- **BUTTON:** Um botão sem ação, onde o usuário tem que definir sua ação.
- **FILE:** Campo de upload de arquivos.
- **HIDDEN:** Campo oculto.

7.2.1. Input Text

O valor TEXT no atributo TYPE de um INPUT indica que o campo será de texto, ou seja, um campo onde o usuário poderá entrar com dados.

Ex: <INPUT TYPE='text' name='login'>

Como sempre crie um arquivo para executar os exemplos da nossa aula e coloque nele um FORM com um INPUT TEXT por enquanto.

Vamos ver três atributos muito utilizados no input text que são o value, o size e o maxlength.

- **Value:** Não é exclusividade do input text, ele faz parte de todos os campos de um formulário, pois este atributo é que contém o valor a ser enviado pelo formulário, podendo já iniciar com um padrão ou não.
- **Size:** O atributo SIZE configura o tamanho do campo e é baseado no número de caracteres. Este valor não limita o tamanho do campo, somente define o tamanho em que será mostrado na página.
- **Maxlength:** Defini a quantidade máxima de caracteres aceitos por este campo.

Exemplo:

```
<FORM>
<p>Entre com seu nome:<INPUT TYPE="TEXT" NAME="Nome"
VALUE="Maria"
SIZE="20" MAXLENGHT="30"></p>
</FORM>
```

Agora adicione estes atributos em nosso campo e faça testes como maxlength =2 normalmente utilizado para UF, mude o size para 1 depois para 2 depois para 50 e veja a diferença, agora coloque um value que informe ao usuário “Digite seu login”.

7.2.2. Input PASSWORD

Para o valor PASSWORD (senha) no atributo TYPE aplicam-se todos os atributos do tipo TEXT exceto que todas as letras digitadas apareçam com um asterisco.

Ex: <INPUT TYPE='password' name='senha' size='30' maxlength='12'>

Lembre-se sempre de adicionar a cada vez às novas funcionalidades que você vai aprendendo.

7.2.3. Input CHECKBOX

Campo de múltipla escolha tem como particularidade o atributo name que tem o conteúdo igual para todas as opções de escolha que existirem no form. Podemos ter mais de uma múltipla escolha no mesmo form com nomes diferentes, mas as opções de cada uma tem o mesmo nome.

```
Ex: <INPUT TYPE='checkbox' name='area' value='programacao'>Programação<br>
    <INPUT TYPE='checkbox' name='area' value='analise'>Análise de Sistemas<br>
    <INPUT TYPE='checkbox' name='area' value='designer'>Design<br>
```

Outro atributo particular do checkbox é o CHECKED que define uma ou todas as opções como marcadas por padrão, pois caso o usuário não preencha nada já teremos algum dado marcado.

7.2.4. Input RADIO

Ao contrário do checkbox este é um campo de escolha única. Teremos várias opções onde apenas uma pode ser selecionada. O padrão do name segue o mesmo que o checkbox.

```
Ex: <INPUT TYPE='radio' name='profissao' value='programador'>Programador<br>
    <INPUT TYPE='radio' name='profissao' value='analista'>Analista de Sistemas<br>
    <INPUT TYPE='radio' name='profissao' value='designer'>Designer<br>
```

Outro atributo similar ao do checkbox é o SELECTED que funciona da mesma forma.

7.2.5. Input SUBMIT, RESET e BUTTON

Podemos falar destes três juntos, pois são iguais com relação a configuração apenas tem funções distintas.

- O **submit** envia os dados do formulário.
- O **reset** restaura os dados padrão do formulário.
- O **button** não tem nenhuma função definida por padrão, o usuário define alguma função para ele.

Ex: `<INPUT TYPE='submit' value='Enviar' name='enviar'> <INPUT TYPE='reset' value='Limpar' name='limpar'>`

Ex: `<INPUT TYPE='button' value='Somar' name='somar'>`

Coloque no final do seu formulário um input SUBMIT e um RESET. Depois preencha o formulário e clique em um dos dois para ver o que acontece. Ao clicar no RESET os dados devem voltar a ser igual a quando entramos na página e quando clicar em SUBMIT deve ser enviado para outra página que está definida no ACTION do form.

7.2.6. Input FILE

Este campo é diferente dos outros, pois ele contém um text e um button juntos onde eles têm uma função específica de procurar um arquivo em seu computador e fazer o upload do mesmo.

Ex: `<INPUT TYPE='file' name='upload'>`

Para que o upload funcione precisamos informar ao formulário (`<FORM>`) que vamos trabalhar com envio de arquivos, essa informação é passada através do atributo enctype com o valor "**multipart/form-data**" o valor padrão do enctype é "text/plain".

7.2.7. Input HIDDEN

O valor HIDDEN no atributo TYPE define dados que devem ser passados ao programa interpretador, embora não estejam visíveis na página.

Geralmente os valores destes campos são atribuídos no momento em que esta sendo feita a consistência dos dados no formulário via uma linguagem de scripts, como JavaScript que estudaremos ainda nessa apostila.

Ex: `<INPUT TYPE='hidden' name='oculto' value='pagina1' >`

Tem apenas os atributos padrões NAME E VALUE.

7.3. Select

Select é um componente muito utilizado para apresentar listas que 1 ou alguns itens serão selecionados, estes itens são representados pela tag `<OPTION>`.

Podendo o usuário escolher mais de uma. A criação de caixas de listagem com múltipla seleção é idêntica a criação de caixas de listagem sem múltiplas seleções com exceção do atributo MULTIPLE que deve ser acrescentado.

Exemplo 1: apenas 1 item pode ser selecionado:

```

<SELECT name='lista' >
    <OPTION value='PE'>Pernambuco</OPTION>
    <OPTION value='CE'>Ceará</OPTION>
</SELECT>
    
```

Exemplo 2: mais de 1 item pode ser selecionado:

Veremos também o atributo SIZE que especifica quantos itens serão mostrados, para selecionar mais de 1 item deve ser pressionada a tecla SHIFT enquanto se clica.

```

<SELECT name='lista' SIZE='3' MULTIPLE >
    <OPTION value='PE'>Pernambuco</OPTION>
    <OPTION value='CE'>Ceará</OPTION>
    <OPTION value='BA'>Bahia</OPTION>
    <OPTION value='RN'>Rio Grande do Norte</OPTION>
</SELECT>
    
```

7.4. TextArea

TextArea é uma área para entrada de texto, onde podemos definir o tamanho de linhas e colunas através dos atributos ROWS e COLS.

Ex: <TEXTAREA rows='20' cols='20' name='texto'>Texto</TEXTAREA>

Como sempre adicione este campo ao seu form para ver o resultado.

8. Desafio

Construa uma página utilizando tudo que você aprendeu até agora, porém a página deve ficar igual a figura abaixo:



9. O que é CSS e como funciona.

CSS sigla para Cascading Style Sheet, que traduzindo significa folha de estilo em cascata.

Consiste num 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 aplicado, como organizar uma coleção de documentos.

Resumindo: CSS é uma linguagem de estilo utilizada para definir a apresentação de documentos web.

9.1. Por que utilizar?

Quando utilizamos CSS temos alguns benefícios bem claros, que justificam o uso do CSS, são estes:

- Controle do layout de vários documentos a partir de uma folha de estilos.
- Utilização de avançadas técnicas de desenvolvimento.
- Facilidade na edição de um layout.

9.2. Efeito Cascata

O efeito cascata nada mais é, do que a definição de uma prioridade para a aplicação de uma regra CSS. Quando existir uma duplicidade de regras para o mesmo elemento, que regra o navegador deve aplicar?

A regra mais importante e de prioridade mais alta é a regra declarada como importante pelo do usuário, em segundo vem a regra declarada como importante pelo desenvolvedor, em terceiro vem à regra declarada pelo desenvolvedor, em quarto vem à regra declarada pelo usuário e por último vem as regras definidas pelo navegador do usuário.

9.3. Sintaxe Básica

Uma folha de estilos é composta por uma ou várias regras CSS, que por sua vez são compostas em sua forma básica por 3 partes: um **seletor**, uma **propriedade** e um **valor**, veja abaixo:

```
SELETOR { PROPRIEDADE: VALOR; }
```

Onde,

- **SELETOR:** Identifica onde (elemento) está regra será aplicada. Podendo ser identificado por sua tag, ID ou classe.
- **PROPRIEDADE:** É o atributo ao qual a regra será aplicada (font, color, background).
- **VALOR:** É o valor propriamente dito a ser aplicado na propriedade selecionada. Observe que se usa dois-pontos (:) e não igual (=) para aplicar um valor a uma propriedade.

9.3.1. Sintax: seletor

Como vimos na página anterior o seletor pode ser uma tag de um elemento HTML, um ID ou uma classe. Mais o que diferencia um do outro?

- Se utilizarmos a tag html como seletor, significa que todos os elementos daquela tag seguirão esta regra.
- Se utilizarmos o ID significa que apenas este elemento seguirá esta regra.
- Se definirmos uma classe, apenas os elementos que forem desta classe seguirão esta regra.

Vamos aos Exemplos:

Seletor TAG:

```
p {  
    font-size: 12px;  
}
```

Esta regra está definindo que todo parágrafo terá o tamanho de fonte de 12 pixels.

Seletor Classe

Agora vamos ver como criar uma classe para um determinado grupo de elementos. Podemos criar classes de duas formas:

1 – Apenas para um tipo de elemento

```
p.fontePequena {  
    font-size: 12px;  
}  
p.fonteGrande {  
    font-size: 20px;  
}
```

Desta forma dentro das tags <p> poderíamos definir que classe ela deve utilizar. Para isto basta adicionar o atributo class desta forma:

<p class="fonteGrande"> ou <p class="fontePequena">

2 – Para qualquer tipo de elemento

```
.textoGrande {  
    font-size: 20px;  
}
```

Observem que neste exemplo não definimos nenhuma tag então podemos aplicar a qualquer elemento html. Sempre utilizando o atributo class.

Observação: Ao utilizar folhas de estilo, devem-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.

9.4. Tipos de CSS

Existem 3 tipos de css, são eles:

- **Inline:** este é escrito dentro do atributo style da tag html.
- **Interno:** que é escrito dentro das tags <style> </style> dentro do <head> do documento html.
- **Externo:** que é escrito em um arquivo separado e apenas referenciado dentro do documento html.

9.4.1. CSS Inline

Este é o menos utilizado de todos pois, perde muitas vantagens do CSS. Ele é aplicado dentro do atributo style da tag, ou seja, para cada tag temos que definir uma regra.

Exemplo:

```
<p style="color:#000000;">  
Um texto escrito na cor preta.  
</p>
```

9.4.2. CSS Interno

Devemos apenas acrescentar a tag <style> dentro do cabeçalho e dentro da tag style declarar todas as regras a serem utilizadas pelo documento. Geralmente aplicado quando temos apenas um arquivo, pois para aplicar as regras em outro arquivo teríamos que copiar e colar.

Exemplo:

```
<head>
<title>Exemplo CSS</title>
<style type="text/css">
.regal {
font-size: 16px;
}
.regra10 {
font-color: #FFFFFF;
}</style>
</head>
```

9.4.3. CSS Externo

O arquivo externo deve conter a extensão **.css** e dentro dele apenas as regras que você deseje declarar. Normalmente aplicado quando temos um site grande contendo vários arquivos.

Exemplo:

```
Arquivo : estilo.css
.regal {
font-size: 16px;
}
...
.regra10 {
font-color: #FFFFFF;
}
```

Para referenciar este arquivo ao documento HTML devemos adicionar uma tag no cabeçalho(<head>) do documento.

Exemplo:

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

9.5. Cores e fundos

9.5.1. Propriedade: Color

A propriedade color representa cor de primeiro plano de um elemento html.

Vamos supor que todos os parágrafos do meu site devem está na cor azul, como ficaria a regra css que deveríamos escrever?

```
p {  
  color: #0000CC;  
}
```

Observação: Cores utilizando padrão RGB;

Um outro exemplo:

```
.minhaClasse {  
  color: #ff0000;  
}
```

Defina uma regra em que todos os Cabeçalhos de nível 1 sejam verdes.

```
h1 {  
  color: #009933;  
}
```

9.5.2. Propriedade: background-color

A propriedade background-color define a cor de fundo de um elemento html.

Caso você queira alterar a cor de fundo de todo seu documento basta adicionar um regra a tag body e definir um background-color.

```
Ex: body {  
  background-color: #00FFFF;  
}
```

Mas podemos usar em outros elementos html como por exemplo no parágrafo, vamos criar uma regra que deixe o texto vermelho e fundo preto de um parágrafo.

```
Ex: p {  
  color: #00FFFF;  
  background-color: #000000;  
}
```

9.5.3. Propriedade: background-image

Se não quisermos utilizar uma cor de fundo e sim uma imagem utilizamos a propriedade background-image.

```
Ex: .bgImagem {  
    background-image: url("imagem.jpg");  
}
```

Onde, "imagem.jpg" é o nome do arquivo que deve está na mesma pasta que o arquivo CSS ou o arquivo html caso esteja utilizando CSS interno ou CSS inline.

Agora crie uma imagem de 100px por 100px e coloque ela como background do elemento BODY do documento html

Perceba que a imagem ficou repetida por toda a extensão do documento.

9.5.4. Propriedade: background-repeat

Vimos anteriormente que quando utilizamos uma imagem como background ela se repete tanto no eixo X quanto no eixo Y.

Agora vamos ver o background-repeat que controla essa repetição ou não permite está repetição.

```
Ex: body{  
    background-image: url("imagem-1.jpg");  
    background-repeat: no-repeat;  
}
```

Os possíveis valores para o background-repeat são:

- **repeat-x**: Repete apenas no eixo X.
- **repeat-y**: Repete apenas no eixo Y.
- **repeat**: Repete apenas nos eixos X e Y.
- **no-repeat**: Não repete a imagem.

9.5.5. Propriedade: background-attachment

A propriedade background-attachment define se a imagem de background vai ser fixa ou vai rolar junto com a tela que é a opção padrão!

```
Ex: body{  
    background-image: url("imagem-1.jpg");  
    background-repeat: no-repeat;  
    background-attachment: fixed;  
}
```

Os possíveis valores para o background-attachment são:

- **fixed**: Deixa o background fixo e não rola com a página.
- **scroll**: Faz com que a imagem role com a página.

9.5.6. Propriedade: background-position

A propriedade background-position define a posição da tela onde a imagem vai ser apresentada, por padrão a imagem é apresentada na posição 0,0 que é o canto esquerdo superior da tela.

Existem várias maneiras de definir o posicionamento da imagem na tela, todas elas usando coordenadas. As coordenadas podem ser escritas em porcentagem ou em unidades fixas (px, cm, etc), podemos também utilizar palavras para definir o posicionamento são elas: top, bottom, center, left e right.

```
Ex: body{
    background-image: url("imagem-1.jpg");
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: 20px 20px;
}
```

```
Ex: body{
    background-image: url("imagem-1.jpg");
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: 20% 20%;
}
```

```
Ex: body{
    background-image: url("imagem-1.jpg");
    background-repeat: no-repeat;
    background-attachment: fixed;
    background-position: top right;
}
```

9.5.7. Propriedade: background

A tag background é um compilador de todas as tags vistas anteriormente, podendo assim escrever em uma só linha todas as propriedades vistas.

A sintax a ser seguida é a seguinte:

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

Caso uma das propriedades não seja declarada ela segue o valor padrão da propriedade.

9.6. Fontes, Textos e Links

9.6.1. Propriedade: Font Family

Com este atributo nós podemos escolher se o texto vai aparecer utilizando a font Arial, Verdana ou outra qualquer.

Lembrando que o computador do usuário tem que ter a fonte para que ela apareça. Por isso podemos definir uma lista de fonts e não apenas uma, está lista também define a prioridade das fontes sendo a primeira a mais importante, caso ela não esteja instalada vai para segunda e assim por diante.

No final da lista, pode ser incluída uma referência a uma família genérica, que será usada caso nenhum dos nomes coincida com o nome de uma fonte do sistema.

```
Ex. 1 : h1 {  
    font-family: arial;  
}  
p {  
    font-family: Verdana, "Times New Roman";  
}
```

9.6.2. Propriedade: Font Style

A propriedade font-style define como a fonte vai aparecer. Se ela vai está em *italic*, *oblique* ou *normal*.

```
Ex: p {  
    font-style: italic;  
}
```

Lembre-se sempre de aplicar os conhecimentos adquiridos nos nossos documentos de testes.

9.6.3. Propriedade: Font Variant

Está é uma propriedade que poucos conhecem, mas pode ser bastante útil, ela tem apenas 2 valores, são eles: normal e small-caps a small-caps são fontes maiúsculas de tamanho reduzido.

Para que o small-caps funcione deve está disponível na máquina do usuário.

```
Ex: p {  
    font-variant: small-caps;  
}
```

9.6.4. Propriedade: Font Weight

Está propriedade define se a fonte vai aparecer normal ou em negrito ou quanto de

negrito vai aparecer. Normalmente se usa normal ou bold como valores para esta propriedade, mas alguns navegadores aceitam uma numeração entre 100-900 com intervalos de 100 em 100 para definir o quão pesada será a fonte.

```
Ex: h2 {  
    font-weight: bold;  
}
```

9.6.5. Propriedade: Font Size

A propriedade font-size indica o tamanho que a fonte deve aparecer, as unidades básicas aceitas por essa propriedade são pixels(**px**) e porcentagem(**%**), podemos também usar pontos(**pt**).

A grande diferença entre essas unidades é que px e pt são absolutas enquanto % pode ser modificada pelo usuário tornando assim seu site mais acessível a pessoas que tem alguma restrição como problemas de visão, monitor de baixa resolução.

```
Ex: p {  
    font-size: 15px;  
}  
  
h1 {  
    font-size: 80%;  
}
```

9.6.6. Propriedade: Font

A propriedade font na verdade é um compilador de todas as propriedades que vimos até agora. A ordem de valores seguida por ele é:

font-style | font-variant | font-weight | font-size | font-family

```
Ex: p {  
    Font: oblique bold 20px Verdana;  
}
```

9.6.7. Propriedade: Text Ident

Na escola vimos que todo parágrafo começa com a primeira linha recuada, e quando temos que escrever uma redação ou algum documento tem que utilizar esta regra. Mas como fazer isso em um documento html?

Para isso utilizamos a propriedade text-indent. Ela representa o tamanho do recuo da primeira linha do parágrafo em pixels.

```
Ex: p {  
    text-indent: 20px;
```

```
}
```

Com esta regra CSS todo <p> terá um recuo de 20px na primeira linha.

9.6.8. Propriedade: Text Align

Como o próprio nome diz o text-align trata do alinhamento do texto, que pode ser **left**, **right**, **center** ou ainda **justify** que este não tínhamos no html.

```
Ex: td {  
    text-align: left;  
}  
Aqui todos os textos de qualquer td ira ficar alinhado a esquerda.  
p {  
    text-align: justify;  
}  
Aqui todos os textos dentro de um <p> vão está justificados.
```

9.6.9. Propriedade: Text Decoration

O text-decoration serve basicamente para “decorar” um texto, colocar um texto como sublinhado ou riscado, por exemplo.

Vou mostrar 3 exemplos aqui, que vocês devem colocar em um documento html para visualizar o efeito.

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

9.6.10. Propriedade: Letter Spacing

O espaçamento entre as letras. Isto já é feito facilmente nos editores de texto, mas como fazer isto em um documento html? Para isto usamos o letter-spacing, para definir o espaço entre os caracteres de um texto.

```
Ex: p {  
    letter-spacing: 2px;  
}
```

```
h2 {  
    letter-spacing: 4px;  
}
```

Neste exemplo temos a distância de 2px entre os caracteres de um parágrafo e de 4px entre os caracteres de um cabeçalho nível 2.

9.6.11. Propriedade: text transform

A propriedade text-transform controla a capitalização do texto, ou seja, controla se o texto vai aparecer maiúsculo ou não, ou quem vai aparecer maiúsculo.

Temos 4 valores aceitos por essa propriedade são eles: capitalize, uppercase, lowercase ou none.

- **Capitalize;** Torna apenas a primeira letra de cada palavra maiúscula.
- **Uppercase:** Torna todo o texto em maiúsculo.
- **Lowercase:** Torna todo o texto em minúsculo.
- **None:** Apresenta o texto da forma que ele foi digitado no arquivo HTML.

```
Ex: p {  
    Text-transform: uppercase;  
}
```

9.6.12. Propriedade: Links

Você pode aplicar tudo que já vimos sobre personalização aos links (cor, fonte, espaçamento e etc). O que vamos poder acrescentar agora são personalizações diferenciadas para cada estado do link por exemplo visitado, não visitado, ativo e outros, para isso vamos utilizar as pseudo-classes.

9.6.12.1. Propriedade: Pseudo-classe

Uma pseudo-classe permite personalizar levando em conta um evento ou estado de uma tag html.

As pseudo-classes que temos para o link são:

- **Link:** Usada para links não visitados.
- **Visited:** Usada para links visitados.
- **Active:** Usada para links ativos.
- **Hover:** Usada quando o cursor do mouse está sobre o link.

9.6.12.2. Exemplos: Link, visited, active e hover

Agora vamos ver alguns exemplos de uso destas pseudo-classes.

Ex: **Removendo o sublinhado do link:**

```
a:link {
text-decoration:none;
}

a:visited {
text-decoration:none;
}

a:active {
text-decoration:none;
}

a:hover {
text-decoration:none;
}
```

Ex: **Mudando a cor do link ao passar o mouse:**

```
a:link {
color: blue;
text-decoration:none;
}

a:visited {
color: blue;
text-decoration:none;
}

a:active {
color: blue;
text-decoration:none;
}

a:hover {
```

```
color: red;
text-decoration:none;
}
```

9.7. Agrupando Elementos - Class e id

9.7.1. Identificando e Agrupando Elementos

Às vezes precisamos personalizar um grupo de tags para isso usamos o **CLASS**.

Outras vezes precisamos personalizar apenas um objeto para isso utilizamos o **ID**.

9.7.2. Agrupando: Class

Para agrupar elementos devemos utilizar uma class, que é definida dentro do css, no arquivo html. Dentro da tag identificamos a que classe este elemento faz parte utilizando o atributo class.

Para definir uma classe dentro do css utilizamos o seguinte padrão .nomeDaClasse e em seguida abrimos as chaves para setar as regras css.

```
Ex : .textoSite {
      Font-size: 15px;
}
```

E dentro das tags de texto do site utilizaremos da seguinte forma:

```
Ex: <p class="textoSite">Texto do site</p>
```

9.7.3. Identificando: Id

Agora vamos ver como criar um estilo para um elemento específico.

Para isto utilizamos o atributo ID que identifica no html um objeto, e dentro do css utilizamos o valor deste id para definir as regras css para este elemento.

```
Ex: <h1 id="titVermelho">Título Vermelho</h1>
```

Para definirmos regras css para este cabeçalho vamos utilizar a seguinte notação: #ID
{ }

```
Ex: #titVermelho{
      Color: red;
}
```

9.8. Agrupando Elementos - Span e Div

9.8.1. Agrupando: Span

A tag `` não tem efeito visual nenhum dentro do documento html, é usada apenas para adicionar efeitos dentro do texto através do css.

Ex html: `texto em negrito`

```
Ex CSS: .negrito {
        font-weight: bold;
    }
```

9.8.2. Agrupando: Div

O div é utilizado para agrupar um ou mais elementos.

A syntax para utilizar a div é igual para utilizar um span. Podemos utilizar class ou id.

```
Ex CSS: .eleDiv{
        Color: red;
    }
```

Ex HTML: `<div class="eleDiv">Aqui vai o conteúdo!</div>`

9.9. Margin, Padding e Bordas

9.9.1. Box Model

Os elementos HTML geram uma caixa em sua volta, está caixa é que chamamos de BOX MODEL.

Está caixa é formada por 3 propriedades do elemento são elas:

- Margin
- Padding
- Border

9.9.1.1. Margin

Margem é a distancia da borda até o Box model de outro elemento ou a margem do documento HTML.

Vamos trabalhar agora as margens do elemento HTML que podem ser 4

(esquerda, direita, superior e inferior).

Para definir a margem de um elemento HTML podemos criar uma regra css para o elemento e dentro dela preencher as seguintes propriedades:

- margin-top: Margem superior
- margin-right: Margem Direita
- margin-bottom: Margem Inferior
- margin-left: Margem Esquerda

Exemplo:

```
<h1>Título</h1>
<p>Frase sobre margin!</p>
```

Para isso vamos aplicar uma margem a esquerda no elemento <p>:

```
<style>
p{
margin-left: 20px;
}
</style>
```

9.9.1.2.Padding

O padding é o espaçamento entro o elemento e sua borda.

Para definir o padding de um elemento devemos fazer igual fizemos para definir a margem.

As propriedades css para o padding são:

- padding-top: Padding superior
- padding-right: Padding Direita
- padding-bottom: Padding Inferior
- padding-left: Padding Esquerda

Siga o mesmo exemplo anterior e aplique essas novas funcionalidades.

9.9.1.3.Bordas

Bordas têm diversas funcionalidades desde separar elementos ou apenas delimitar o tamanho de um elemento.

Veremos a partir de agora as seguintes propriedades das bordas:

- **Border-width**
- **Border-color**
- **Border-style**
- **Border**

9.9.1.3.1. Border-width

O border-width define a largura da borda que pode ser determinada por thin, medium, e thick (fina, média e grossa) ou em pixels.

9.9.1.3.2. Border-color

O Border color define a cor que vai ser apresentada na borda.

A cor pode ser definida das seguintes formas: "#000FFF", "rgb(123,123,123)" ou "blue".

9.9.1.3.3. Border-style

Border-style define o estilo de borda que vai ser apresentada.

Podemos utilizar os seguintes estilos:

- **dotted**
- **dashed**
- **solid**
- **double**
- **groove**
- **ridge**
- **inset**
- **outset**

Agora podemos criar um exemplo e testar a variação de estilos:

9.9.1.3.4. Border

Border é um compilador de todas as propriedades vistas anteriormente.

Vamos agora re-escrever a regra css do cabeçalho em uma linha:

```
<style>
h1 {
    border: thick solid blue;
}
</style>
```

O border segue a seguinte ordem:

- **width**
- **style**
- **color**

9.10. Altura Largura e Float

9.10.1. Largura: Width

Width é a propriedade que representa a largura de um objeto.

Quando precisamos aumentar ou diminuir lateralmente um elemento utilizamos dentro de uma regra CSS associada ao elemento a propriedade width.

Está propriedade aceita como valores um número em pixel ou uma porcentagem, exemplo:

Width: 200px;

Ou

Width: 20%

9.10.2. Altura: height

A propriedade height representa a altura de um elemento HTML.

Está propriedade é similar a width e aceita como valores tanto um valor em pixels como uma porcentagem.

Uma observação importante é que normalmente para montar layouts utilizamos valores fixos em pixels tanto na altura como na largura para que nosso layout seja apresentado da mesma forma independente da resolução utilizada pelo usuário.

9.10.3. Flutuando Elementos

Podemos flutuar elementos a esquerda ou a direita utilizando a propriedade float. Isto significa que o elemento vai ser deslocado para direito ou esquerda do documento ou do container que ele estiver.

9.10.4. Clear

A propriedade Clear serve para controlar como os elementos depois de um float vão se comportar. O padrão é que o elemento seguinte ao float ocupe o espaço livre deixado pelo elemento flutuado. No exemplo acima vimos que o texto ficou do lado da imagem e embaixo.

A propriedade clear aceita os seguintes valores: left, right, both ou none.

Normalmente se utiliza o both, que define que a margem superior do elemento vai está abaixo da margem inferior do elemento flutuado.

Um exemplo legal , adicionar uma regra css para a tag HTML <p>

CSS:

```
p{
  clear: both;
}
```

Agora aplique esta nova regra css no exemplo e veja o efeito causado.

9.11. Posicionando Elemento e Layer

9.11.1. Sistema de Coordenadas

Para podermos posicionar um elemento precisamos primeiro entender como vamos dizer onde este elemento vai ser apresentado. O navegador tem um sistema de coordenadas que é representado em pixels, ou seja, tem dois eixos um eixo X na horizontal e um eixo Y na vertical. O canto superior esquerdo representa o ponto 0,0 como é mostrado na imagem abaixo.

Para posicionar um elemento temos que utilizar algumas propriedades são elas:

- **Position:** Que aceita dois valores 'absolute' ou 'relative'.
- **Top ou botton:** Que representa o eixo Y, variando apenas se ele vai medir de cima pra baixo ou de baixo pra cima.
- **Left ou Right** Que representa o eixo X, variando apenas se ele vai iniciar da esquerda ou da direita.

Exemplo:

HTML:

```
<h1>Cabeçalho</h1>
```

CSS:

```
H1 {
```

```
Position: absolute;  
Top: 200px;  
Left: 100px;  
}
```

9.11.2. Position: absolute

Um elemento posicionado como a propriedade position setada como absolute não gera nenhum espaço vazio no documento HTML, depois de ser reposicionado.

Vamos a prática, entenda o exemplo:

HTML:

```
<p> aqui vai texto aqui vai texto aqui vai texto aqui vai texto aqui vai texto<br />aqui  
vai texto aqui vai texto aqui vai texto aqui vai texto aqui vai texto </p>  
<img src='imagem.jpg' />  
<p> aqui vai texto aqui vai texto aqui vai texto aqui vai texto aqui vai texto<br />aqui  
vai texto aqui vai texto aqui vai texto aqui vai texto aqui vai texto </p>
```

CSS:

```
img {  
position: absolute;  
top: 100px;  
left: 100px;  
}
```

9.11.3. Position: relative

Quando posicionamos um elemento utilizando a posição relativa, a posição final do elemento vai ser calculada a partir da posição do elemento no documento.

Com isso ele deixa um espaço vazio no documento, que é o lugar que aquele elemento estava.

Para observarmos melhor isto basta pegar o Exemplo anterior visualizar ele sem CSS com o position absolute como já foi visto e agora com position relative.

9.11.4. Layers

Layers são camadas, que se sobrepõem, então temos a noção de profundidade. As layers são organizadas com por uma numeração onde os números maiores estão na frente(em cima) dos números menores.

Por exemplo a layer de número 1 está na frente da layer número 0, e a layer número 2 está na frente da número 1.

9.11.5. Z-Index

Para organizarmos a numeração das layers utilizamos a propriedade z-index.

Exemplo:

```
#layer1 {  
    position: absolute;  
    left: 10px;  
    top: 50px;  
    z-index: 1;  
}  
#layer2 {  
    position: absolute;  
    left: 50px;  
    top: 10px;  
    z-index: 2;  
}
```

9.12. Web-standard e validação

9.12.1. O que são web-standards?

WEB-STANDARDS são padrões e orientações desenvolvidos pelo W3C que regem o desenvolvimento, tendo como resultado final sites com:

- **Códigos simples e objetivos**
- **Separados em camadas**
- **Multi-plataforma**
- **Melhor relacionado com sites de busca.**

9.12.2. Porque usar?

Quando utilizamos web-standards temos algumas vantagens como:

- **Acessibilidade**
 - Maior Velocidade

- Apresentado da mesma forma em várias plataformas
- Otimização para motores de busca
- **Redução de custo**
 - Diminui a quantidade de retrabalho
 - Diminui o consumo de banda

Essas são algumas das vantagens em se trabalhar com web-standards.

9.12.3. O que precisa para usar web-standards?

Para criar sites utilizando web-standards é necessário está antenado com:

- **As normas do W3C**
- **Boa prática no HTML**
- **Boa prática no CSS**
- **Desenvolvimento em camadas**
- **Validação W3C**

9.12.4. Camadas

Temos basicamente duas camadas quanto tratamos de web:

- **Front-end**
- **back-end**

A camada de front-end é onde fica o HTML, CSS e JavaScript onde:

- O HTML cria a estrutura
- O CSS cria a apresentação
- O JS cria a ação

Na camada de back-end podemos ter várias tecnologias diferentes como:

- **PHP**
- **ASP .NET**
- **Java**

Entre outras.

10. O JavaScript

Javascript é uma linguagem de script poderosa que pode ser usada para aumentar a interatividade de suas páginas Web. Os códigos estão abertos pelo navegador, e pode ser observado nas mais diversas páginas espalhadas pela Web, no qual você poderá copiar e alterar com um mínimo de esforço, se você entendê-los.

10.1. Onde colocar os scripts

Os scripts devem ser colocados dentro das tags `<SCRIPT>` no código HTML, poderá ser inserido em qualquer parte do documento, mas preferencialmente, deve se colocar entre as tags `<HEAD>`. Mais você pode usá-lo entre as tags `<BODY>`

Para iniciar uma tag `<SCRIPT>` , indicaremos sempre o MIME type da linguagem em uso. O MIME type do javascript é “text/javascript”. Por exemplo:

```
<script type="text/javascript">  
</script>
```

10.2. O elemento `<noscript>`

Caso o browser não suporte Javascript ou que ele esteja desabilitado, é uma boa pratica criar uma mensagem de aviso para o usuário da possibilidade de algumas funcionalidades da sua página deixar de funcionar. Por Exemplo:

```
<noscript>  
  <h1>Esse browser não suporta javascript ou está desativado.</h1>  
</noscript>
```

10.3. Diferença entre Javascript e Java

Apesar de terem nomes quase iguais, Javascript e Java não têm quase nada em comum. Enquanto Java é uma poderosa linguagem de programação, desenvolvida e comercializada pela Sun Microsystems, o JavaScript é uma linguagem de scripts, criada pela Netscape, e que depende de um programa para ser interpretada, no caso o browser. O que gera muita confusão é que além dos nomes serem parecidos, a sintaxe tanto do Javascript quanto do Java foram trazidas das linguagens C e C++.

10.4. Variáveis

Variáveis são containeres que armazenam temporariamente um determinado valor. O javascript é uma linguagem case sensitive, isto é, ele é sensível às diferenças entre letras maiúsculas e minúsculas. Outro detalhe muito importante é que não podemos colocar espaços ou outras pontuações em variáveis a menos que sejam as permitidas.

Variáveis podem ser qualquer palavras, desde que :

- Comece com uma letra
- Utilize o sublinhado(underscore) em palavras compostas
- E que não seja uma palavra reservada da linguagem.

Exemplos de variáveis:

```

var animal = "Leão"; // variável valida
var Animal = "Vaca"; // variável valida
var 1animal = "Cobra"; // variável invalida
var _animal = "Macaco"; // variável valida
var else = "Passaro"; // variável invalida é uma palavra reservada
    
```

10.5. Palavras Reservadas

Palavras reservadas são palavras chaves usada pela linguagem para a execução de determinadas instruções. Essas palavras não podem ser usadas como variáveis. Veja a tabela das palavras reservadas do javascript:

abstract	float	public
boolean	for	return
break	function	short
byte	goto	static
case	if	super
catch	implements	switch
char	import	synchronized
class	in	this
const	instanceof	throw
continue	int	throws
default	interface	transient
do	long	true
double	native	try
else	new	var
extends	null	void
false	package	while
final	private	with
finally	protected	

10.6. Operadores Matemáticos

Freqüentemente, você irá realizar cálculos matemáticos em scripts. Veja a tabela a baixo:

Operador	O que será feito
X + z (Números)	Será somado o valor de x com o valor de z
X + z (String)	Será concatenado a string x com a string z
x * z	Será multiplicado o valor de x com o valor de z
X /z	Será dividido o valor de x com o valor de z
X % z	Modulo (resto) da divisão de x com o valor de z
X++	Usa a variável x e depois incrementa com 1
++x	Incrementa a variável x com 1 e depois utiliza a variável.
X--	Usa a variável x e depois decrementa com 1
X--	decrementa a variável x com 1 e depois utiliza a variável.
-x	Inverte o sinal de x

Exemplo Usando operadores Matemáticos:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>matematicos</title>
</head>
<body>

<script type="text/javascript">
    var z = 100;
    var y = 10;
    var resultado = 0;
    resultado = z + y;
```

```

document.write(resultado);
document.write("<br>");
</script>
</body>
</html>
    
```

10.7. Operadores Lógicos - Atribuições

Ao inserir um valor numa variável, você está atribuindo um valor a uma variável, e para isso estará utilizando um operador de atribuição. Exceto o sinal de igual, os outros operadores servem para modificar o valor das variáveis. Veja a tabela abaixo:

Atribuição	O que será feito
$X=z$	Define que o valor de x é igual ao valor de z
$X+=z$	O valor de x será igual ao valor de x + z. Seria o mesmo que fazer ($x = x + z$)
$X-=z$	O valor de x será igual ao valor de x - z. Seria o mesmo que fazer ($x = x - z$)
$X*=z$	O valor de x será igual ao valor de x * z. Seria o mesmo que fazer ($x = x * z$)
$x/=z$	O valor de x será igual ao valor de x / z. Seria o mesmo que fazer ($x = x / z$)
$X\%=z$	O valor de x será igual ao valor de x % z. Seria o mesmo que fazer ($x = x \% z$)

10.8. Operadores Lógicos – Comparações

Freqüentemente, você irá comparar o valor de uma variável com outra, ou o valor de uma variável com uma string ou um número. Veja a tabela a baixo:

comparação	O que será feito
$X == z$	Será verdadeiro (true) caso x e z forem iguais
$X != z$	Será verdadeiro (true) caso x e z forem diferentes
$x > z$	Será verdadeiro (true) caso x for maior que z
$x >= z$	Será verdadeiro (true) caso x for maior ou igual a z
$X < z$	Será verdadeiro (true) caso x for menor do que z
$X <= z$	Será verdadeiro (true) caso x for menor ou igual a z
$x \&\& z$	Será verdadeiro (true) se x e z forem verdadeiro (true)

X z	Será verdadeiro (true) se x ou z forem verdadeiro (true)
!x	Será verdadeiro se x for falso

10.9. Comentando o script

Toda linguagem de programação possui formas de comentar o código e vale lembrar que é uma boa prática de programação comentar. Embora seja muito fácil achar códigos mal comentados ou praticamente sem comentários em códigos. Então quando for codificar lembre-se sempre que até você mesmo depois de alguns dias, meses ou anos poderá ter esquecido como programou tal função e será necessário ler e reler algumas vezes o código para entendê-lo.

Exemplo de comentário em Javascript:

```
// Comentário de uma linha
/*
    Comentário para mais de uma linha
*/
```

10.10. Comandos Condicionais e de Repetição

10.10.1. A condição if

A condição if é utilizada para realizar testes, caso seja passada um valor true então será possível entrar no bloco de código do if, caso contrário não.

Abaixo a sintaxe para criar scripts com a condição if:

```
if([condição]){
    // o código para ser executado ficará aqui
}
```

Exemplo:

```
<script type="text/javascript">
    var valor = 90;
    if(valor < 100){
        document.write("Estoque a baixo de 100");
    }
</script>
```

10.10.2. Usando else

O ELSE é o caso contrário, ele é utilizado para indicar um novo bloco de comando caso a condição do IF não seja satisfeita.

Abaixo a sintaxe para criar scripts com a condição if:

```
if([condição]){  
    // o código para ser executado ficará aqui  
}  
else{  
    // será executado esse trecho de código caso a condição seja falsa.  
}
```

Exemplo:

```
<script type="text/javascript">  
  
    var valor = 120;  
    var string = "limão";  
  
    if((valor > 100) && (string == "laranja")){  
        document.write("Estoque de laranja dentro do limite");  
    }else{  
        document.write("Estoque de limão dentro do limite");  
    }  
  
</script>
```

10.10.3. Operador ternário

É Possível realizar um bloco de if em apenas uma linha, da seguinte forma:

(condição? Parte verdadeira : Parte falsa)

Ideal para situações onde não exijam textos muito longos. A isso damos o nome de operador ternário.

Como você pode ver, a primeira essa expressão é muito menor do que um bloco de if normal, mais deve se alertar sobre o seu uso. Se usada em excesso, pode tornar seu código de difícil compreensão e comprometer sua funcionalidade, principalmente se você começar a aninhar varias dessas operações entre si.

Exemplo

```
<script type="text/javascript">
```

```
var valor = 20;
var retorno = "";
(valor < 100 ? retorno = valor * 5 : retorno = valor / 5 )
document.write(retorno);
```

</script>

10.10.4. Comandos de repetição

10.10.4.1. Repetição com FOR

A declaração for contém três partes, separadas por ponto e vírgula (;).O primeiro contém uma instrução (de uma série de instruções separadas por um ponto e vírgula) que será executada uma vez antes do ciclo já começado. A segunda contém uma condição que é verificada no início de cada iteração do ciclo, e uma terceira instrução (ou, mais uma vez, um conjunto de instruções separados por vírgula) que é executado ao final de cada iteração.

A sintaxe do for é a seguinte:

```
for(i=0; i<10; i++){
    // o bloco de código que será repetido ficará aqui.
}
```

Exemplo

```
<script type="text/javascript">
    for(var i=0; i<10; i++){
        document.write(i);
        document.write("<br/>");
    }
</script>
```

10.10.4.2. Repetição com WHILE

O WHILE é um dos mais simples controladores iterativos ele permite repetir uma série de operações até que avalia a condição verdadeira (TRUE)

A sintaxe do while é:

```
while([condição]){
    // o bloco de código que será repetido ficará aqui.
}
```

Exemplo:

```
<script type="text/javascript">
    var i=0;
```

```
while(i<10){
    document.write(i);
    document.write("<br/>");
    i++;
}
```

</script>

10.10.4.3. Repetição com DO...WHILE

O controle é realizado no final de cada iteração significado que, mesmo que a condição de nunca for verdadeira (true), o conteúdo do loop será executado pelo menos uma vez.

A sintaxe do DO...WHILE é a seguinte:

```
do{
    // Bloco de código ficará aqui.
}while([condição]);
```

Exemplo:

<script type="text/javascript">

```
var i = 0;
do {
    document.write( i );
    document.write( "<br>" );

    i++;
} while (i < 10);
```

</script>

10.10.4.4. Repetição com FOR...IN

Uma variação interessante do FOR é o FOR...IN. Esse loop nos permite varrer vários elementos de um objeto.

<script type="text/javascript">

```
var propriedades;

for(propriedades in window){
    document.write(propriedades);
    document.write("<br>");
}
```

</script>

10.10.4.5. Controle de LOOP

JavaScript suporta declarações que modificam o controle do fluxo. São elas a **break** e **continue**. Elas são frequentemente usadas em loops. A declaração **break** irá encerrar um loop antes do seu término, enquanto o **continue** irá pular a interação do looping atual. Veja os exemplos abaixo:

Exemplo 1:

```
<script type="text/javascript">

    for(var i=0; i< 10; i++){
        if(i == 5){
            break;
        }
        document.write(i);
        document.write("<br>");
    }
</script>
```

Exemplo 2:

```
<script type="text/javascript">

    for(var i=0; i< 10; i++){
        if(i == 5){
            continue;
        }
        document.write(i);
        document.write("<br>");
    }
</script>
```

10.10.5. Eventos

Os eventos em javascript vieram para suprir uma necessidade do HTML, já que a linguagem HTML só permite o click num link e enviar a página para outro link. O JavaScript vai então acrescentar muitos eventos ao HTML.

Os eventos Javascript, associados às funções, aos métodos e aos formulários, abrem uma grande porta para uma verdadeira interatividade das páginas.

10.10.5.1. Tabela de Eventos

A tabela abaixo mostra os diferentes eventos suportados por JavaScript:

Eventos	Quando ocorre
Click	Quando o utilizador clica sobre um botão, um link ou outros elementos.
Load	Quando a página é carregada pelo browser.
Unload	Quando o utilizador saia da página.

MouseOver	Quando o utilizador coloca o ponteiro do mouse sobre um link ou outro elemento.
MouseOut	Quando o ponteiro do mouse sai de cima de um link ou outro elemento.
Focus	Quando um elemento de formulário tem o focus, isto é, que está ativo.
Blur	Quando um elemento de formulário perde o focus, isto é, quando o deixa de estar ativo.
Change	Quando o valor de um campo de formulário é modificado.
Select	Quando o utilizador seleciona um campo dentro do elemento do formulário.
Submit	Quando o utilizador clica sobre o botão Submit para enviar um formulário.

10.10.5.2. Gestão de Eventos

Para que um evento ocorra de forma eficaz, precisamos associar um evento a uma ação definida por você. Desde uma função ou um alerta na tela. A sintaxe é:

```
onClick="[função]"
```

Por exemplo:

```
onClick="alert('Exemplo de click');"
```

Nesse exemplo, ao clicar será exibida uma caixa de dialogo com o texto **“Exemplo de click”**.

Varios exercicios resolvidos com essas funções serão apresentados no final da apostila.

10.10.6. Mensagens e Funções

10.10.6.1. Alert

O primeiro tipo de mensagem que vamos ver é o ALERT que serve apenas para alertar o usuário para alguma coisa, nesta mensagem não temos nenhum retorno do usuário, ele apenas lê e da Ok para fechar o quadro.

Sua syntax é simples:

```
Alert('mensagem');
```

Este tipo de mensagem é muito utilizada para informar erros de preenchimentos de formulários, respostas positivas ou negativas sobre o envio de algum dado ao servidor, entre outras coisas.

10.10.6.2. Confirm

Agora veremos o CONFIRM que tem uma característica diferente, ele tem dois botões um de OK outro de CANCEL onde recebemos no JavaScript TRUE se o usuário clicar no OK e FALSE caso ele clique no CANCEL.

A sua sintax é bem parecida com a do alert:

```
Confirm('mensagem');
```

Normalmente utilizamos dentro de um IF para verificar quais dos botões foi clicado.

10.10.6.3. Prompt

O prompt é o tipo de mensagem que podemos fazer perguntas mais complexas ao usuário como por exemplo: Qual seu nome?

No prompt aparece a pergunta e um campo de texto para que o usuário responda a mesma.

Sua sintax é um pouco mais complexa que as outras:

```
Variável = prompt('Mensagem', 'Texto Input');
```

- **Variável** É a variável que vai receber o valor digitado pelo usuário.
- **Mensagem** É a mensagem propriamente dita ao usuário.
- **Texto Input** É o texto que aparece inicialmente dentro do campo de texto.

Exemplo:

```
Nome = prompt('Qual seu nome?', 'Digite aqui!');
```

```
Alert('Seja bem vindo '+Nome);
```

10.10.6.4. Função

Função é um bloco de código que só será executado quando for chamado, podendo ou não ter um retorno.

Existem dois tipos de função:

- **Funções da Linguagem** São funções que estão embutidas na linguagem.
- **Funções do Usuário** São funções criadas pelo programador.

10.10.6.5. Sintax

Para criarmos uma função temos que seguir uma sintax que é bem simples:

```
Function Nome da função ( Parâmetros )
```

```
{
```

```
  Bloco de código
```

```
  Return 'valor';
```

}

Onde,

- **Function** É a palavra reservada que declara a função.
- **Nome da função** É o nome pelo qual vamos acionar a função.
- **Parâmetros** São valores que vamos passar para a função que podem ser separados por ‘,’.
- **Bloco de código** Como o próprio nome diz é o local onde vamos colocar todo o código a ser executado.
- **Return** Opicional que gera um retorno para quem chamou a função.

10.10.6.6. Funções da Linguagem

Vamos mostrar apenas algumas funções que estão embutidas dentro da linguagem:

- - **eval** Função que calcula o conteúdo de uma string.
- - **parseInt** Converte uma String em um Inteiro
- - **parseFloat** Converte uma String em um ponto flutuante
- - **isNaN** Retorna se é um número ou não, caso contrário retorna True.

10.10.7. Array's e String's

10.10.7.1. Array

Arrays ou Vetores são um conjunto de variáveis, normalmente do mesmo tipo de dados que são acessados através de um índice.

Existem três formas bem simples de instanciar um Array:

Primeira:

```
Var meuArray = new Array();
```

Desta forma a variável meuArray agora é um array sem limite de posições, onde você pode ir colocando valores e adicionando posições automaticamente.

Segunda:

```
Var meuArray2 = new Array(5);
```

Agora o nosso array tem 5 posições e não podem que não podem ser ultrapassadas.

Terceira:

```
Var meuArray3 = new Array('Laranja', 'Limão', 'Mamão');
```

Desta forma nosso Array já foi instanciado com valores para as posições 0, 1 e 2.

10.10.7.2. Acessando um Array

Para acessar um array usamos a seguinte sintax:

```
NomeDoArray[ índice ];
```

Então se eu quiser Atribuir a uma variável o valor da soma do primeiro com o segundo valor de um array utilizaria o seguinte código.

```
Var soma = meuArray[0] + meuArray[1];
```

10.10.7.3. String

A classe String que é o tipo de dados utilizado para armazenar textos em variáveis. A classe String possui muitos métodos e propriedades, sem dúvida é uma das classes mais completas do javascript. Para criarmos uma variável do tipo string

basta usar a seguinte sintax:

```
Var meuTexto = "uma string";
```

Ou podemos usar o operador new.

```
Var meuTexto = new String();
```

Ou ainda

```
Var meuTexto = new String("uma string");
```

Propriedade e os principais métodos da classe string:**Propriedade:**

- **length** : Está propriedade informa a quantidade de caracteres de uma string.

Métodos:

- **charAt(índice)** Retorna o caractere da posição indicada pelo índice.
- **indexOf(caractere, apartir)** Retorna o índice do primeiro caractere encontrado igual ao que foi passado no primeiro parâmetro, o segundo parâmetro é opcional, serve para informar a partir de que posição deve começar a busca.
- **toLowerCase()** Coloca toda a string em minúscula.

- **toUpperCase()** Coloca toda a String em maiúscula.
- **substr(inicio, fim)** Devolve um pedaço da string que está localizado entre o caractere inicio e o caractere fim.
- **replace(existente, nova)** Substitui a string existente pela nova string onde ela aparecer dentro a string.

10.10.8. Datas

10.10.8.1. Trabalhando com datas

JavaScript pode pegar a data e a hora do computador e depois trabalhar de diversas formas com esses dados. Com objeto do tipo Date você possui vários métodos para controlar datas e horas dentro do seu script. Descobrimos a data e hora você poderá armazená-las, fazer cálculos com datas ou até convertê-las em strings.

10.10.8.2. Criando um objeto date

Podemos declarar e iniciar um objeto date das seguintes formas:

1. Declarando um objeto date novo, sem passar nenhum valor:

```
var data = new Date();
```

2. Declarando um objeto date novo, passando um valor em milissegundos, contando desde 1 de janeiro de 1970 às 00:00:00 GMT. No exemplo abaixo será criado a data referente ao dia 06 de Outubro de 2008 às 21:32

```
var data = new Date(1223339561248);
```

3. Declarando um objeto date novo, passando uma data ou uma data e hora em formato de string:

```
var data = new Date("1 October 2008");
```

4. Declarando um objeto date novo, passando a data e hora separadas por vírgula, no seguinte formato:

```
var data = new Date(2008, 9,12,22,30,50,10);
```

Nesse último caso, será criada a data 12 de Outubro de 2008 às 22:30:50 e 10 milissegundos.

Observação: veja que você passa os valores na seguinte ordem: ano, mês, dia, hora, minuto, segundo e milissegundos. Caso precise você poderá passar a data e ignorar o restante dos parâmetros.

10.10.8.3. Recuperando a data do sistema

A seguir temos uma tabela informando os métodos do objeto Date e que será muito útil para manipulação de datas do seu computador através do browser:

Método	Descrição
getDate()	O dia do mês
getDay()	O dia da semana como um inteiro, onde domingo será considerado 0, segunda-feira 1 e assim por diante.
getMonth()	O mês como um inteiro, onde Janeiro será considerado 0, Fevereiro 1 e assim por diante.
getFullYear()	O ano em formato de quatro dígitos.
getDateString()	Retorna a data completa baseada na corrente time zone como uma string humanamente legível. Por exemplo, "Sun Oct 12 2008".

10.10.8.4. Como inserir o dia da semana

Para exemplificar iremos criar um script onde tragam por extenso o dia da semana atual.

1. Crie um documento HTML novo e dentro da tag <BODY> criaremos uma array contendo todos os dias da semana.

```
var arrDias = new Array("Domingo", "Segunda", "Terça", "Quarta",  
"Quinta", "Sexta", "Sabado");
```

2. Depois iremos imprimir na tela, e acessar o dia da semana passando o dia atual com a função getDay().

```
var data = new Date();  
document.write("Hoje é "+arrDias[data.getDay()]);
```

3. Teste no seu browser.

10.10.8.5. Alterando o objeto date

Também é possível mostrar para o usuário a hora atual do sistema. Para isso usamos os métodos abaixo:

Método	Descrição
getHours()	Retorna a hora
getMinutes()	Retorna os minutos
getSeconds()	Retorna os segundos

10.10.9. Interação com Usuário

10.10.9.1. Criando POPUP

Criar uma POPUP em JavaScript é bastante simples, basta chamarmos a propriedade `open()` do objeto `window`. Como no exemplo abaixo:

```
<script type="text/javascript">
    window.open("exercicio1.html","Janela","width=400,height=400");
</script>
```

Como vimos anteriormente, ao chamarmos `window.open()`, estamos passando também três parâmetros. Vamos detalhar cada um deles agora.

- Primeiro parâmetro - Link da página que será exibida na POPUP
- Segundo parâmetro - Alias ou título da POPUP. É útil caso você abra uma POPUP e depois tente abrir novamente a mesma POPUP. Ele irá carregar na mesma janela.
- Terceiro parâmetro – as propriedades da POPUP. veja a tabela de propriedades

Propriedade	Descrição
status	Usado para informar se o popup irá possuir ou não a barra de status. Podendo receber os valores “no” ou “0”, “yes” ou ou “1”
toolbar	Usado para informar se o popup irá possuir ou não a barra de ferramentas. Podendo receber os valores “no” ou “0”, “yes” ou ou “1”
menubar	Usado para informar se o popup irá possuir ou não a barra de menu. Podendo receber os valores “no” ou “0”, “yes” ou ou “1”
top	Distância em pixel do topo de onde o popup será iniciando
left	Distância em pixel da esquerda de onde o popup será iniciando
width	Largura em pixel da popup
height	Tamanho em pixel da popup
scrollbars	Usado para informar se o popup irá possuir ou não a barra de rolagem. Podendo receber os valores “no” ou “0”, “yes” ou ou “1”
maximized	Usado para informar se o popup irá possuir ou não a opção de maximizar. Podendo receber os valores “no” ou “0”, “yes” ou ou “1”
resizable	Usado para informar se o popup irá possuir ou não a possibilidade de modificar seu tamanho. Podendo receber os valores “no” ou “0”, “yes” ou ou “1”

fullscreen	Usado para informar se o popup irá possuir ou não opção de tela cheia . Podendo receber os valores “no” ou “0”, “yes” ou ou “1”
------------	--

Todas as propriedades serão passadas como string separadas por vírgula.

10.10.9.2. Chamada de funções na POPUP

Ao abrirmos uma POPUP podemos fazer chamadas de funções da janela principal (a que nós usamos para abrir a POPUP). Veja o exemplo abaixo:

Crie um arquivo HTML chamado janelaPai.html e copie o código abaixo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Exemplo</title>
</head>

<body>
<script type="text/javascript">
    function mensagem(msg){
        alert(msg);
    }
</script>
<h1>Janela Pai</h1>
<input type="button" name="botao" value="Botão" onclick="window.open('popup.html');">
</body>
</HTML>
```

10.10.9.3. Inserindo ID no elemento

A maioria das tags HTML possui a propriedade ID, com um elemento possuindo um ID nós poderemos acessar e trabalhar com todas as suas propriedades através do JavaScript.

Por exemplo:

```
<input type="button" name="botao" id="botao" value="Botão"/>
```

Uma observação importante que devemos ter é que cada página não poderá ter vários IDs repetidos, pois se possuir perderemos a identificação do elemento, isso quer dizer que, cada elemento possuirá um ID único e distinto do resto da página.

10.10.9.4. Acessando um elemento

Para acessarmos um elemento além de definirmos um id para o elemento, precisaremos informar para o javascript qual elemento queremos manipular. Para isso usamos a propriedade `document.getElementById()`, passando como parâmetro o id do elemento que queremos acessar:

```
document.getElementById("botao") // acessando o elemento input que é um botão.
```

10.10.9.5. Alterando estilo

Podemos alterar estilos de elementos através de javascript, veja o exemplo abaixo.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Exemplo</title>
</head>

<body>
<script type="text/javascript">

    function cor(){
        document.getElementById("nome").style.color = "RED";
    }
</script>
<span id="nome" style="font-size:20px; color:#0000FF">Educandus</span>
<input type="button" name="mudar" value="Mudar Cor" onclick="cor()">
</body>
</html>
```

10.10.9.6. Validando um formulário

Para validarmos um formulário, usaremos o mesmo método `getElementById()` para acessar o elemento e verificamos se foi digitado algo. Veja o exemplo abaixo:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Exemplo</title>
</head>

<body>
<script type="text/javascript">

    function validar(){
        if(document.getElementById("nome").value == ""){
            alert("Por favor, preencha o campo nome");
        }
    }
</script>
</body>
</html>
```

```
        }else{
            document.getElementById("frm").submit();
        }
    }
</script>
<form name="frm" id="frm" method="get" action="pagina2.html">
<input type="text" name="nome" id="nome" value=""/>
<input type="button" name="validar" value=" Validar" onclick="validar()">
</form>
</body>
</html>
```

10.10.10. Funções e Definições

10.10.10.1. Funções de String

Agora veremos uma lista de funções para trabalhar com String. Lembrando que estas funções são parte do objeto string e deve existir um objeto instanciado para que elas possam ser utilizadas.

- **charAt(índice)** Retorna um caractere da string, índice é a posição do caractere a ser retornado.
- **charCodeAt(índice)** Retorna o valor ASCII do caractere, da posição informada no índice.
- **concat (s2,s3,...,sN)** Concatena duas ou mais strings.
- **fromCharCode(char1,char2,...,charN)** Retorna uma string a partir dos valores ASCII, informados nos parâmetros char1,char2,...,charN.
- **indexOf(str,[índiceInicial])** Retorna a posição da primeira ocorrência de uma string em outra.
- **lastIndexOf(str,[índiceInicial])** Retorna a posição da última ocorrência de uma string em outra. Neste caso a pesquisa é feita no sentido inverso ou seja da direita para esquerda.
- **Match(exp)** Executa uma expressão regular em uma string.
- **replace(existente, nova)** Substitui a string existente pela nova string onde ela aparecer dentro a string.
- **Slice(inicio,[fim])** Extrair parte de uma string. Iniciando no índice indicador por incio e terminando no índice indicado por fim
- **Split(separador)** Separar uma string em uma array de strings, o parâmetros separador indica que caractere vai servir de limite para separação da string.
- **toLowerCase()** Coloca toda a string em minúscula.
- **toUpperCase()** Coloca toda a String em maiúscula.
- **substr(inicio, fim)** Devolve um pedaço da string que está localizado entre o caractere inicio e o caracter fim.

10.10.10.2. Funções de Array

Agora veremos uma lista de funções para trabalhar com array:

- **Join(separador)** Retorna uma string com todos os elementos separados pelo separador passado por parâmetro.
- **length** Retorna o número de elementos.
- **Pop()** Retorna e remove o último elemento.
- **Push()** Adiciona elementos no final.
- **Reverse()** Inverte a ordem.
- **Shift()** Retorna e remove o primeiro elemento.
- **Slice(início, [fim])** Retorna parte do array, iniciando o corte no índice informado pelo parâmetro início, e terminando o corte no índice informado pelo parâmetro fim ou no final do array.
- **toString()** Retorna uma string, formada pelos elementos do array.
- **Unshift()** Adiciona elementos no início.

10.10.10.3. Funções de Matemáticas

Agora vamos ver uma lista de funções matemáticas:

- **Math.abs(número)** → retorna o valor absoluto do número (ponto flutuante)
- **Math.ceil(número)** → retorna o próximo valor inteiro maior que o número
- **Math.floor(número)** → retorna o próximo valor inteiro menor que o número
- **Math.round(número)** → retorna o valor inteiro, arredondado, do número
- **Math.pow(base, expoente)** → retorna o cálculo do exponencial
- **Math.max(número1, número2)** → retorna o maior número dos dois fornecidos
- **Math.min(número1, número2)** → retorna o menor número dos dois fornecidos
- **Math.sqrt(número)** → retorna a raiz quadrada do número
- **Math.SQRT2** → retorna a raiz quadrada de 2 (aproximadamente 1.414)
- **Math.SQRT_2** → retorna a raiz quadrada de $\frac{1}{2}$ (aproximadamente 0.707)
- **Math.sin(número)** → retorna o seno de um número (ângulo em radianos)
- **Math.cos(número)** → retorna o cosseno de um número (ângulo em radianos)
- **Math.acos(número)** → retorna o arco cosseno de um número (em radianos)
- **Math.tan(número)** → retorna a tangente de um número (ângulo em radianos)
- **Math.atan(número)** → retorna o arco tangente de um número (em radianos)
- **Math.pi** → retorna o valor de PI (aproximadamente 3.14159)
- **Math.log(número)** → retorna o logaritmo de um número
- **Math.E** → retorna a base dos logaritmos naturais (aproximadamente 2.718)
- **Math.LN2** → retorna o valor do logaritmo de 2 (aproximadamente 0.693)
- **Math.LOG2E** → retorna a base do logaritmo de 2 (aproximadamente 1.442)
- **Math.LN10** retorna o valor do logaritmo de 10 (aproximadamente 2.302)
- **Math.LOG10E** → retorna a base do logaritmo de 10 (aproximadamente 0.434)

11. EXERCÍCIOS RESOLVIDOS

1. Quais programas são usados para criar uma página html?

Resposta:

São utilizados desde simples editores de texto como o Bloco de Notas, Word Pad, Word, até sofisticados aplicativos destinados a criação de páginas como o FrontPage, Dream Weaver, etc, que possibilitam a utilização diversas ferramentas do tipo "drag and drop" que economizam bastante tempo para a criação de páginas. Além destes, existem vários outros que podem ser baixados pela Internet, contudo vale a pena ressaltar que uma página html é composto de texto plano (ASCII) e deve ser salva com htm ou html.

2. Que comando devo inserir (e onde) para criar três linhas em branco entre duas linhas já existentes?

Comando
, que além de ser usado para quebrar uma linha também é usado para criar uma linha em branco. Neste caso ele deve ser inserido três vezes entre as duas linhas.

linha a linha a linha a linha...

linha b linha b linha b linha....

3. Crie um texto e aplique os 4 tipos de alinhamento (align="right", "left", "center", justify")

Alinhamento de texto à direita:

Direita Direita Direita Direita Direita Direita Direita Direita Direita Direita Direita Direita
Direita

Alinhamento de texto à esquerda:

Esquerda Esquerda Esquerda Esquerda Esquerda Esquerda Esquerda Esquerda Esquerda Esquerda
Esquerda Esquerda Esquerda

Alinhamento de texto centralizado:

Centralizado Centralizado Centralizado Centralizado Centralizado Centralizado
Centralizado Centralizado Centralizado Centralizado

Alinhamento de texto justificado:

Justificado Justificado Justificado Justificado Justificado Justificado Justificado
Justificado Justificado Justificado Justificado Justificado Justificado Justificado

4. Qual é o comando de cabeçalho que insere o texto com o maior tamanho de fonte?

É o comando `<h1>`

5. Qual é o comando de cabeçalho que insere o texto com o menor tamanho de fonte?

É o comando `<h6>`

6. Quais são os comandos mínimos html necessários para criar uma página web?

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

7. Quais são as tags responsáveis por uma lista numerada e uma lista não numerada?

Listas numeradas: ``

Listas não numeradas: ``

8. Quais são os tipos de numerações possíveis de serem aplicadas em listas ordenadas?

1 | A | a | I | i

9. Para que servem os comandos `<DT>`, `<DL>`, e `<DD>` ?

`<DL>` Abre uma lista de definição

`<DT>` Identifica o termo

`<DD>` Define o termo

10. Quais são os tipos de marcadores possíveis de aplicar em uma lista não numerada?

Square | Disk | Circle

11. Como você resolveria a seguinte lista abaixo descrita?

O código seria o seguinte:

```
<OL>
<LI></LI>
<LI></LI>
<LI></LI>
<OL>
<LI></LI>
<LI></LI>
</OL>
<LI></LI>
<OL>
<LI></LI>
<LI></LI>
</OL>
</OL>
```

O resultado apresentado seria:

1. Primeiro item da lista
2. Segundo item da lista
3. Terceiro item da lista
 1. Primeiro item do terceiro item da lista
 2. Segundo item do terceiro item da lista
4. Quarto item da lista
 1. Primeiro item do terceiro item da lista
 2. Segundo item do terceiro item da lista
 3. Terceiro item do terceiro item da lista

12. Codifique a definição de listas conforme abaixo

FAQ

Acrônimo de Frequently asked Questions. Documento que reúne respostas às perguntas feitas com maior frequência pelos usuários de um determinado sistema. Para cada assunto na internet, existe uma FAQ correspondente;

FIREWALL

Sistema de segurança que evita que a rede de computadores de uma determinada empresa seja acessada por estranhos através da internet.

O código seria o seguinte:

```
<DL>
<DT></DT>
<DD></DD>
<DT></DT>
```

```
<DD></DD>  
</DL>
```

13. Faça o arquivo css que modifique a cor de textos.

```
<html>  
  
<head>  
<style type="text/css">  
h1 {color: #00ff00}  
h2 {color: #dda0dd}  
p {color: rgb(0,0,255)}  
</style>  
</head>  
  
<body>  
<h1>This is header 1</h1>  
<h2>This is header 2</h2>  
<p>This is a paragraph</p>  
</body>  
  
</html>
```

14. Defina a margem direita de uma imagem utilizando um valor %:

```
<html>  
<head>  
<style type="text/css">  
img  
{  
position:absolute;  
right:5%  
}  
</style>  
</head>  
<body>  
<h1>This is a Heading</h1>  
  
<p>Some text.</p>  
</body>  
</html>
```

15. Configure o layout de uma tabela usando css:

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



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

17. Crie um página sem tabelas usando css:

```
<html>  
<head>  
<style type="text/css">  
div.container  
{  
width:100%;  
margin:0px;  
border:1px solid gray;  
line-height:150%;  
}  
div.header,div.footer  
{  
padding:0.5em;  
color:white;  
background-color:gray;  
clear:left;  
}  
h1.header  
{  
padding:0;  
margin:0;  
}  
div.left  
{  
float:left;  
width:160px;  
margin:0;  
padding:1em;  
}  
div.content  
{  
margin-left:190px;  
border-left:1px solid gray;  
padding:1em;  
}  
</style>  
</head>  
<body>  
  
<div class="container">  
<div class="header"><h1 class="header">W3Schools.com</h1></div>
```

```
<div class="left"><p>"Never increase, beyond what is necessary, the number of entities
required to explain anything." William of Ockham (1285-1349)</p></div>
<div class="content">
<h2>Free Web Building Tutorials</h2>
<p>At W3Schools you will find all the Web-building tutorials you need,
from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.</p>
<p>W3Schools - The Largest Web Developers Site On The Net!</p></div>
<div class="footer">Copyright 1999-2005 by Refsnes Data.</div>
</div>

</body>
</html>
```

18. Faça a primeira letra ficar especial em qualquer texto usando css:

```
<html>
<head>
<style type="text/css">
p:first-letter
{
color: #ff0000;
font-size:xx-large
}
</style>
</head>

<body>
<p>
You can use the :first-letter pseudo-element to add a special effect to the first letter of a
text!
</p>
</body>

</html>
```

19. Defina o estilo de quatro bordas usando css:

```
<html>
<head>
<style type="text/css">
p.dotted {border-style: dotted}
p.dashed {border-style: dashed}
p.solid {border-style: solid}
p.double {border-style: double}
p.groove {border-style: groove}
p.ridge {border-style: ridge}
</style>
</head>
</html>
```

```
p.inset {border-style: inset}
p.outset {border-style: outset}
</style>
</head>

<body>
<p class="dotted">A dotted border</p>

<p class="dashed">A dashed border</p>

<p class="solid">A solid border</p>

<p class="double">A double border</p>
</body>

</html>
```

20. Defina o tamanho da fonte usando css:

```
<html>
<head>
<style type="text/css">
h1 {font-size: 150%}
h2 {font-size: 130%}
p {font-size: 100%}
</style>
</head>

<body>
<h1>This is header 1</h1>
<h2>This is header 2</h2>
<p>This is a paragraph</p>
</body>

</html>
```

21. Faça uma página HTML que contenha dois campos textos, nome e e-mail. Fazer uma função javascript que valide o e-mail digitado pelo usuário. O e-mail tem que conter "@".

```
<form name="form1" action="" method="post" onsubmit="return valida_dados(form1)">
<p><label>Nome:</label></p>
<p><input type="text" name="nome" /></p>
<p><label>E-mail:</label></p>
<p><input type="text" name="email" /></p>
<p><input type="submit" name="enviar" value="Enviar" /></p>
</form>
```

```
<script type="text/javascript" language="javascript">

function valida_dados (form1){

if (form1.nome.value=="")

{
alert ("Por favor informe o seu nome!");

return false;
}

if (form1.email.value==" " || form1.email.value.indexOf('@', 0) == -1 || form1.email.value.indexOf('.', 0) ==
-1)

{
alert ("O formato do endereço de email é inválido!");

return false;
}

return true;
}

</script>
```

22. Escreva o código JS que escreve o resto da palavra antes de terminar de digitar:

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Auto Completar</title>
</head>
<script>
fruits = new Array('cleber','mamao','melancia','mercadoria','marmelada');
names = new Array('tom','dick','harry','john','petter','foo','bar');
function autocomplete(n,ac_array){
if (n.value == "") return 0;
if (event.keyCode == 8 && n.backspace){
n.value = n.value.substr(0,n.value.length-1);
n.backspace = false;
}

var r = n.createTextRange();
tmp= n.value;
if (tmp == "")return 0;
for (z=0;z<ac_array.length;z++){
tmp2 = ac_array[z];
count = 0;
for (i = 0;i<tmp.length;i++){
if (tmp2.charAt(i) == tmp.charAt(i)){
count++
}
}
}
if (count == tmp.length){
diff = tmp2.length - tmp.length;
if (diff <= 0) break;
```

```

    kap = "";
    for (i=0;i<tmp2.length;i++){
    if (i >= tmp.length) kap += tmp2.charAt(i);
    }
    n.backspace = true;
    r.text += kap;
    r.findText(kap,diff*-2);
    r.select();
    return 0;
    }
    }
    n.backspace = false;
    return 0;
    }
</script>
<body>
Frutas: <input name='fruit' type='text' class='textbox' title="Opening"
onkeyup="autocomplete(this,fruits)" size="20">
<br /><i><font size="2">Exemplo: 'cleber','mamao','melancia','mercadoria','marmelada'</font></i>
<hr />
Nomes: <input name='Name' type='text' class='textbox' title="Opening"
backspace='false' onkeyup="autocomplete(this,names)" size="20">
<br /><i><font size="2">Exemplo: 'tom','dick','harry','john','petter','foo','bar'</font></i>
</body>
</html>

```

23. Faça um código JavaScript que insira e retire elementos de uma página html:

```

<html>
<head>
<title>Incluindo e removendo elementos</title>

<script language="JavaScript">

function append(form)
{
    if ( form.input.value )
    {
        var newItem = document.createElement("LI");
        newItem.appendChild(document.createTextNode(form.input.value));
        document.getElementById("myUL").appendChild(newItem);
    }
}

function replace(form)
{
    if (form.input.value)
    {
        var newItem = document.createElement("LI");
        var lastChild = document.getElementById("myUL").lastChild;
        newItem.appendChild(document.createTextNode(form.input.value));
        document.getElementById("myUL").replaceChild(newItem, lastChild);
    }
}

```

```

    }

    function restore()
    {
        var oneChild;
        var mainObj = document.getElementById("myUL");
        while ( mainObj.childNodes.length > 2 )
        {
            oneChild = mainObj.lastChild;
            mainObj.removeChild(oneChild);
        }
    }
}
</script>
</head>

<body bgcolor=lightGreen>

Esta é a lista de itens:
<ol ID="myUL">
<li>Primeiro Item</li>
<li>Segundo Item</li>
</ol>
<form>
Entre um texto para incluir ou substituir na lista:
<INPUT TYPE="text" NAME="input" SIZE=30>
<p>
<INPUT TYPE="button" VALUE="Incluir na Lista"  onClick="append(this.form)">
<INPUT TYPE="button" VALUE="Trocar o último item"  onClick="replace(this.form)">
<INPUT TYPE="button" VALUE="Restaurar a Lista"  onClick="restore()">
</p>
</form>
</body>
</html>

```

24. Faça um código JavaScript que mude a cor de fundo da página html, ao usuário clicar em um link indicando a mudança:

```

<html><head>

    <title>Muda cor de fundo da página</title>
    <script language="javascript">
function dataModif()
{
    mes = new Array("Jan", "Fev", "Mar",
                    "Abr", "Mai", "Jun",
                    "Jul", "Ago", "Set",
                    "Out", "Nov", "Dez");

    modif = new Date(document.lastModified);
    document.write("<h3>Última modificação nesta página ",
                    modif.getDate(), "/",
                    mes[modif.getMonth()], "/",
                    modif.getFullYear(), "</h3>");
}
function mudaCor(cor)
{

```

```

    document.bgColor = cor;
}
</script>
</head><body bgcolor="yellow">
    <h1 align="center">Utilizando o Objeto Document</h1>
    <script language="javascript">dataModif();</script><h3>Última modificação nesta página
27/Jul/2001</h3>
    <h2>Mudar a Cor do Fundo</h2>
    <a href="javascript:mudaCor('white')">Branco</a><br>
    <a href="javascript:mudaCor('lightGreen')">Verde</a><br>
    <a href="javascript:mudaCor('lightblue')">Azul</a><br>
    <a href="javascript:mudaCor('yellow')">Amarelo</a><br>
    <script language="javascript">
if (document.referrer)
    document.write("<h3>Referenciada por '", document.referrer, "'</h3>");
</script><h3>Referenciada por 'http://equipe.nce.ufrj.br/joao/webmaster/exemplos/'</h3>
</body></html>

```

25. Faça um código JavaScript que emita uma alert no instante em que uma combobox for marcada e também no momento em que for desmarcada:

```

<HTML>
<HEAD>
<TITLE>Exemplo de checkbox</TITLE>

<SCRIPT>
function clicou(campo)
{
    if (campo.checked)
        alert("O campo está selecionado");
    else
        alert("Campo desmarcado !");
}
</SCRIPT>

</HEAD>

<BODY BGCOLOR=beige>
<H1>Selecionando uma checkbox</H1>

<FORM>
<INPUT TYPE=checkbox onclick="clicou(this)"> Marque esta opção !<BR>
</FORM>

</BODY>
</HTML>

```

26. Faça um código JavaScript que emita capture o evento de passar o mouse por cima de uma imagem e mostre hints dessas imagens:

```

<HTML>
<HEAD>
<TITLE>Captura do evento MouseOver</TITLE>

```

```

<SCRIPT LANGUAGE="javascript">

personagem = new Array("pernalonga", "patolino", "frangolino",
                        "bip-bip", "coiote", "tasmânia");

function entrei(n)
{
    document.saida.texto.value = personagem[n];
}

function sai(n)
{
    document.saida.texto.value = "";
}

function nada()
{
}

</SCRIPT>

</HEAD>
<BODY BGCOLOR="#FFC800">
<CENTER>
<H1>Evento onMouseOver</H1>

<A href="javascript:nada()" onMouseOver="entrei(0)" onMouseOut="sai(0)">
<IMG SRC="../imagens/image1.gif" NAME=img1 BORDER=0 width="50" height="50"></A>
<A href="javascript:nada()" onMouseOver="entrei(1)" onMouseOut="sai(1)">
<IMG SRC="../imagens/image2.gif" NAME=img2 BORDER=0 width="50" height="50"></A>
<A href="javascript:nada()" onMouseOver="entrei(2)" onMouseOut="sai(2)">
<IMG SRC="../imagens/image3.gif" NAME=img3 BORDER=0 width="50" height="50"></A>
<A href="javascript:nada()" onMouseOver="entrei(3)" onMouseOut="sai(3)">
<IMG SRC="../imagens/image4.gif" NAME=img4 BORDER=0 width="50" height="50"></A>
<A href="javascript:nada()" onMouseOver="entrei(4)" onMouseOut="sai(4)">
<IMG SRC="../imagens/image5.gif" NAME=img5 BORDER=0 width="50" height="50"></A>
<A href="javascript:nada()" onMouseOver="entrei(5)" onMouseOut="sai(5)">
<IMG SRC="../imagens/image6.gif" NAME=img6 BORDER=0 width="50" height="50"></A>
<FORM NAME=saida>
<INPUT TYPE=TEXT NAME=texto>
</FORM>
</CENTER>
</BODY>
</HTML>
    
```

27. Faça um código css para editar as propriedades da tag body que determine uma imagem ao fundo:

```

BODY {
background : #C0D9D9 url(images/fundo.gif) repeat;
font : 8pt Verdana, Geneva, Arial, Helvetica, sans-serif;
color : #666666;
margin : 20px 0px 20px 0px;
text-align: center;
}
    
```

28. Crie um código css e html que impeça que uma página possa ser impressa pelo botão imprimir:

Página HTML

```
<html>
<head>
<link href="estilos.css" rel="stylesheet" type="text/css" media="print">
</head>
<body>
<div class="naover">

... conteúdo da página

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

Folha estilos: estilos.css

```
.naover {
visibility: hidden;
}
```

29. Crie um menu dinâmico usando css:

```
<style type="text/css">
#menu div.barraMenu,
#menu div.barraMenu a.botaoMenu {
font-family: sans-serif, Verdana, Arial;
font-size: 8pt;
color: white;
}

#menu div.barraMenu {
text-align: left;
}

#menu div.barraMenu a.botaoMenu {
background-color: #556975;
color: white;
cursor: pointer;
padding: 4px 6px 2px 5px;
text-decoration: none;
}

#menu div.barraMenu a.botaoMenu:hover {
```

```
background-color: #637D4D;
}
```

```
#menu div.barraMenu a.botaoMenu:active {
background-color: #637D4D;
color: black;
}
</style>
```

```
<div id="menu"><div class="barraMenu">
<a class="botaoMenu" href="">Opção 1</a>
<a class="botaoMenu" href="">Opção 2</a>
<a class="botaoMenu" href="">Opção 3</a>
<a class="botaoMenu" href="">Opção 4</a>
</div></div>
```

background-color de a.botaoMenu : cor de estado repouso (out).
background-color de a.botaoMenu:hover : cor de estado sobre (over).
background-color de a.botaoMenu:active : cor de estado selecionado.

30. Cria a funcionalidade em css que decora um campo select do formulário:

```
option {font-family: verdana; font-size: 10px; color: white}
```

```
option.uno {background-color: #CCC}
option.dos {background-color: #666}
```

```
<select>
<option class="um">Opcao</option>
<option class="dois">Opcao</option>
<option class="um">Opcao</option>
<option class="dois">Opcao</option>
<option class="um">Opcao</option>
<option class="dois">Opcao</option>
</select>
```

31. Use css para colocar um texto que possa ser lido na vertical em nosso documento html:

```
<style>
#textovertical {writing-mode: tb-rl; filter: flipv fliph}
</style>
```

32. Use css para arredondar as esquinas de caixa na sua página:

O código HTML :

```
<div class="caixaacima">
<div class="cajaabaixo">
Lorem ipsum dolor sit amet, consectetur
...
</div>
</div>
```

Agora vemos o código CSS.

```
.caixaacima {
width: 600px;
background-image: url("arriba.gif");
background-position: top center;
background-repeat: no-repeat;
background-color: #660000;
color: #ffffff;
}
.caixaabaixo {
background-image: url("abajo.gif");
background-position: bottom left;
background-repeat: no-repeat;
padding: 5px 5px 5px 5px;
}
```

33. Usando JavaScript, crie um código que poderemos modificar e selecionar o valor da propriedade action.

```
<script language="javascript">
```

```
function enviar(form)
```

```
{
```

```
    if((boletim.inscrever.checked == true) && (boletim.cancelar.checked == true))
```

```
        { alert("Por favor, marque somente um campo"); return true; }
```

```
    if((boletim.inscrever.checked == false) && (boletim.cancelar.checked == false))
```

```
        { alert("Deve indicar se deseja se inscrever ou cancelar"); return true; }
```

```
    if (boletim.inscrever.checked == true)
```

```
        { boletim.action = "exemplos/inscrever.asp"; }
```

```
    if (boletim.cancelar.checked == true)
```

```
{ boletim.action = "exemplos/cancelar.asp"; }  
  
form.submit()  
  
}  
  
</script>
```

O formulário...

```
<form name="boletin" method="GET" action="exemplos/inscrever.asp" target="_blank">  
  
<b>Inscrição gratuita ao boletim de novidades</b><br>  
  
Por favor, introduza seu e-mail:  
  
<input type="text" name="email" size="20"><br>  
  
<input type="checkbox" name="inscrever" value="ON"> Fazer <b>inscrição</b> |  
  
<input type="checkbox" name="cancelar" value="ON"> Cancelar<br>  
  
<input type="button" onClick="enviar(this.form)" value="Enviar">  
  
</form>
```

34. Usando JavaScript, crie um código que O campo "chave" (password) estará bloqueado se o campo "usuario" (text) se encontrar vazio e se desbloqueará em caso contrário.

```
<script language="javascript">  
function bloqDesbloq()  
{  
a = login.usuario.value  
  
if (a != "") { a = true; }  
else { a = false; }  
if (a == true) { login.chave.disabled = false; }  
else { login.chave.disabled = true; }  
}  
</script>
```

O formulário...

```
<form name="login" method="POST" action="pagina_de_login.asp" target="_blank">  
Usuario: <input type="text" name="usuario" size="10" onKeyUp="bloqDesbloq()"><br>  
Chave: <input type="password" name="chave" size="10" disabled>  
</form>
```

35. Temos uma caixa de texto com um valor padrão. Quando o valor introduzido em uma caixa é digitado o texto é "reproduzido" em outro campo, faça essa funcionalidade usando javascript:

O Script...

```
<script linguagem="javascript">  
  
function passaValor(form)  
  
{ exemplo2.campo2.value = exemplo2.campo1.value; }  
  
</script>
```

O formulário...

```
<form name="exemplo2" method="POST">  
  
Seu nome: <input type="text" name="campo1" onKeyUp="passaValor(this.form)"><br>  
  
Nome introduzido: <input type="text" name="campo2" ReadOnly>  
  
</form>
```

36. Crie uma caixa elegante de forma que essa caixa se destaque em relação a pagina usando html:

```
<table width="280" cellspacing="1" cellpadding="3" border="0" bgcolor="#1E679A">  
<tr>  
  <td><font color="#FFFFFF" face="arial, verdana, helvetica">  
<b>Caixa curiosa com HTML</b>  
  </font></td>  
</tr>  
<tr>  
  <td bgcolor="#ffffcc">  
  <font face="arial, verdana, helvetica">  
  Este é o interior da caixa. Esperamos que seja elegante... é muito simples.  
  </font>  
  </td>  
</tr>  
</table>
```

37. Construa um código HTML em etiquetas META para atualizar uma página ou fazer um re-encaminhamento a outra página automaticamente.

```
<html>  
<head>  
<title>Re-endereçar o navegador a outra URL</title>
```

```

<META HTTP-EQUIV="REFRESH" CONTENT="5;URL=http://www.criarweb.com">
</head>
<body>
Esta página muda em 5 segundos pelo portal de CriarWeb.com
</body>
</html>
    
```

38. Construa uma Página que se atualiza sozinha ao passar uns segundos:

```

<html>
<head>
<title>Atualizar a URL</title>
<META HTTP-EQUIV="REFRESH" CONTENT="1;URL=refrescar.html"> </head>
<body>
Hora:
<script>
minhaData = new Date()
document.write(minhaData.getHours() + ":" + minhaData.getMinutes() + ":" +
minhaData.getSeconds())
</script>
</body>
</html>
    
```

39. Construa uma página web com tabelas:

```

<table width="778" cellspacing="1" cellpadding="3" border="0" bgcolor="#000000"
align="center">
<tr>
<td width=180 align=center bgcolor="#ffffff"></td>
<td bgcolor="#ffffff" align=center></td>
</tr>
<tr>
<td colspan=2 bgcolor="#ffffff" background="fundohorizontal.gif">
<font face="Garamond">Portada | <a href="#">Noticias | Agenda | Artistas | Buscador | Comunidad |
Tienda</font>
</td>
</tr>
</table>
    
```

40. Construa um html que utilize um desenho para tabelas, não deixando assim a tabela totalmente quadrada:

```

<table width="150" cellspacing="0" cellpadding="0" border="0" bgcolor="B9B9B9">
<tr>
<td style="padding-top:8px; padding-left:9px; padding-right:3px;">
<b>Opções</b>
<br>
<br>
+ O que seja
<br>
+ Opção legal
<br>
+ Mais links
<br>
+ Isto é outro texto
<br>
+ Pêras
<br>
+ Maçãs
<br>
+ Pêssegos
<br>
<br>
</td>
</tr>
<tr>
<td></td>
</tr>
</table>
    
```

41. Faça uma lista em html usando tabelas:

```

<table cellpadding="2" cellspacing="2">
<tr>
<td valign=top></td>
<td>Elemento 1 da lista</td>
</tr>
<tr>
<td valign=top></td>
<td>Este seria um segundo elemento</td>
</tr>
<tr>
<td valign=top></td>
<td>Característica adicional a ressaltar</td>
</tr>
<tr>
<td valign=top></td>
<td>Pode haver elementos cujas características ocupem várias linhas. Colocamos VALIGN=TOP na célula do bullet para que apareça acima.</td>
</tr>
</table>
    
```

42. Escrever por completo uma declaração de frames sem bordas:

```
<html>
<head>
  <title>Definição de Frames</title>
</head>
<frameset cols="200,*" border="0" frameborder="0" e framespacing="0">
  <frameset rows="170,*">
    <frame src="pagina1.html">
    <frame src="pagina2.html">
  </frameset>
  <frame src="pagina3.html">
</frameset>
</html>
```

43. Retire o feio sublinhado que apresentam os links de uma página web, usando html:

```
<a href="http://www.educandus.com" style="text-decoration:none"> Educandus</a>
```

44. Faça em java script uma função para habilitar e desabilitar todos os campos de um checkbox:

```
function selecionar_tudo(){
  for (i=0;i<document.f1.elements.length;i++)
    if(document.f1.elements[i].type == "checkbox")
      document.f1.elements[i].checked=1
}

function deselecionar_tudo(){
  for (i=0;i<document.f1.elements.length;i++)
    if(document.f1.elements[i].type == "checkbox")
      document.f1.elements[i].checked=0
}

<form name="f1">
Nome: <input type="text" name="nome">
<br>
<input type="checkbox" name="ch1"> Opcao 1
<br>
<input type="checkbox" name="ch2"> Opcao 2
<br>
<input type="checkbox" name="ch3"> Opcao 3
<br>
<input type="checkbox" name="ch4"> Opcao 4
<br>
//Outro campo de formulario:
<select name="outro">
<option value="1">Selecao 1
<option value="2">Selecao 2
</select>
```

```
<br>
<input type="submit">
<br>
<br>
<a href="javascript:selecionar_todo()">Marcar todos</a> |
<a href="javascript:deselecionar_todo()">Marcar nenhum</a>
</form>
```

45. Usando JavaScript faça uma função para desabilitar radio butons:

```
<html>
<head>
  <title>Exemplo para desabilitar radio butons</title>

  <script>
  indice_marcado = 0
  function desabilitar(formulario,idradio){
    formulario.meuradio[indice_marcado].checked = true
    formulario.meuradio[idradio].blur()
  }
  </script>
</head>

<body>
<h1>Exemplo para desabilitar radio butons</h1>

<form name="f1">
<input type="radio" name="meuradio" value="O que for" onclick="desabilitar(this.form,0)"
checked> Olá pessoal!
<br>
<input type="radio" name="meuradio" value="outra coisa" onclick="desabilitar(this.form,1)">
Aqui estamos
<br>
<input type="radio" name="meuradio" value="mais coisas" onclick="desabilitar(this.form,2)"> E
você aí?
</form>

</body>
</html>
```

46. Faça usando JavaScript uma função para Comprovar se as senhas são iguais:

```
<html>
<head>
<title>Validar se a senha e a repetição da senha são iguais</title>
<script>
function comprovarSenha(){
  senha1 = document.f1.senha1.value
  senha2 = document.f1.senha2.value

  if (senha1 == senha2)
```

```
        alert("As duas senhas são iguais...\nRealizariamos as açoes do caso positivo")
    else
        alert("As duas senhas são diferentes...\nRealizariamos as açoes do caso negativo")
    }
</script>
</head>

<body>

<h1>Validar se a senha e a repetição da senha são iguais</h1>
<br>
Escreva uma senha duas vezes, uma em cada campo, e clique o botão. Javascript lhe dirá se as
duas são iguais.

<br>
<form action="" name="f1">
Contrassenha: <input type="password" name="senha1" size="20">
<br>
Repita contrassenha: <input type="password" name="senha2" size="20">
<br>
<input type="button" value="Comprovar se são iguais" onClick="comprovarSenha()">

</form>

</body>
</html>
```

47. Faça um java script que recebe uma data de nascimento e devolve o número de anos desde a data, ou seja, a idade.

```
//calcular a idade de uma pessoa
//recebe a data como um string em formato portugues
//devolve um inteiro com a idade. Devolve false em caso de que a data seja incorreta ou maior que o dia
atual
function calcular_idade(data){

    //calculo a data de hoje
    hoje=new Date()
    //alert(hoje)

    //calculo a data que recebo
    //descomponho a data em um array
    var array_data = data.split("/")
    //se o array nao tem tres partes, a data eh incorreta
    if (array_data.length!=3)
        return false

    //comprovo que o ano, mes, dia são corretos
    var ano
    ano = parseInt(array_data[2]);
    if (isNaN(ano))
        return false
```

```
var mes
mes = parseInt(array_data[1]);
if (isNaN(mes))
    return false

var dia
dia = parseInt(array_data[0]);
if (isNaN(dia))
    return false

//se o ano da data que recebo so tem 2 cifras temos que muda-lo a 4
if (ano<=99)
    ano +=1900

//subtraio os anos das duas datas
edad=hoje.getYear()- ano - 1; //-1 porque ainda nao fez anos durante este ano

//se subtraio os meses e for menor que 0 entao nao cumpriu anos. Se for maior sim ja cumpriu
if (hoje.getMonth() + 1 - mes < 0) //+ 1 porque os meses comecam em 0
    return idade
if (hoje.getMonth() + 1 - mes > 0)
    return idade+1

//entao eh porque sao iguais. Vejo os dias
//se subtraio os dias e der menor que 0 entao nao cumpriu anos. Se der maior ou igual sim que ja cumpriu
if (hoje.getUTCDate() - dia >= 0)
    return idade + 1

return idade
}
```

48. Faça uma função JavaScript que oculte um endereço de correio, que é colocado no mailto de um link, para evitar que seja detectado pelos robôs que buscam endereços em webs para enviar spam.

```
<script language="JavaScript">
usuario="pedro"
dominio="qualquerum.com"
conector="@

function dar_correio(){
    return usuario + conector + dominio
}

function escreve_link_correio(){
    document.write("<a href='mailto:' + dar_correio() + '>' + dar_correio() + '</a>")
}
</script>

<a href="mailto:correio@meudominio.com">correio@meudominio.com</a>

<body>
<!-- em qualquer parte do corpo da página -->
```

```
<script>escreve_link_correio()</script>
</body>
```

49. Crie uma Função em Javascript para a inserção de datas:

```
function IsNumeric(valor)
{
var log=valor.length; var sw="S";
for (x=0; x<log; x++)
{ v1=valor. substr(x,1);
v2 = parseInt(v1);
//Comprovo se é um valor numérico
if (isNaN(v2)) { sw= "N";}
}
if (sw=="S") {return true;} else {return false; }
}

var primeiroslap=false;
var segundoslap=false;
function formateadata(data)
{
var long = data.length;
var dia;
var mes;
var ano;

if ((long>=2) && (primeiroslap==false)) { dia=data.substr(0,2);
if ((IsNumeric(dia)==true) && (dia<=31) && (dia!="00")) { data=data.substr(0,2)+"/"+data.substr(3,7);
primeiroslap=true; }
else { data=""; primeiroslap=false;}
}
else
{ dia=data.substr(0,1);
if (IsNumeric(dia)==false)
{data="";}
if ((long<=2) && (primeiroslap=true)) {data=data.substr(0,1); primeiroslap=false; }
}
if ((long>=5) && (segundoslap==false))
{ mes=data.substr(3,2);
if ((IsNumeric(mes)==true) &&(mes<=12) && (mes!="00")) { data=data.substr(0,5)+"/"+data.substr(6,4);
segundoslap=true; }
else { data=data.substr(0,3);; segundoslap=false;}
}
else { if ((long<=5) && (segundoslap=true)) { data=data.substr(0,4); segundoslap=false; } }
if (long>=7)
{ ano=data.substr(6,4);
if (IsNumeric(ano)==false) { data=data.substr(0,6); }
else { if (long==10){ if ((ano==0) || (ano<1900) || (ano>2100)) { data=data.substr(0,6); } } }
}

if (long>=10)
{
data=data.substr(0,10);
dia=data.substr(0,2);
```

```
mes=data.substr(3,2);
ano=data.substr(6,4);
// Ano nao bisexto e é fevereiro e o dia é maior a 28
if ( ( ano%4 != 0 ) && ( mes ==02 ) && ( dia > 28 ) ) { data=data.substr(0,2)+"/"; }
}
return (data);
}
```

50. Realizar um script em uma página, com Javascript, que faz com que se recarregue constantemente e assegura que não se obtém a página sempre desde o cache.

```
function aleatorio(inferior,superior){
    numPossibilidades = superior - inferior
    aleat = Math.random() * numPossibilidades
    aleat = Math.floor(aleat)
    return parseInt(inferior) + aleat
}
```

```
minhaData = new Date()
dato_url = minhaData.getYear().toString() + minhaData.getMonth().toString() +
minhaData.getDate().toString() + minhaData.getHours().toString() + minhaData.getMinutes().toString() +
minhaData.getSeconds().toString()
```

```
setTimeout("window.location='pagina.html?parametro="+ dato_url + "'", num_aleatorio * 1000)
```

12. Exercício Proposto HTML – CSS -JS

1. Crie uma folha de estilos, chame-a de basico.css, e a carregue no arquivo StyleTest.html.
2. Nesta folha de estilos, usando o mínimo de declarações possível, declare:
 - a) que todo H1 tenha fonte Tahoma, ou sans-serif, se Tahoma não existir
 - b) que todo o texto do corpo (BODY) do arquivo tenha tamanho 10 pontos
 - c) que todos os H1, H2 e H3 sejam vermelhos
 - d) que os H1 tenham tamanho 24 pontos
 - e) que os H2 tenham tamanho 18 pontos
 - f) que os H3 tenham tamanho 14 pontos
3. Mude a cor do fundo da página para azul marinho (navy) e a cor default do texto para bran-co em uma única declaração.
4. Faça com que todo texto marcado em itálico apareça em azul ciano (cyan).
5. Carregue a folha de estilos basico.css em outros arquivos HTML e veja o que acontece. Faça com que uma dessas outras páginas tenha uma cor de fundo clara (amarela, por e-xemplo) e cor de texto escuro (diferente daquela definida por basico.css) sem que isto alte-re as outras paginas que usam o mesmo arquivo.
6. Faça com que o primeiro parágrafo do arquivo StyleTest.html tenha texto verde.
7. Faça com que a célula do meio da tabela de StyleTest.html tenha texto vermelho sobre fundo amarelo (a tabela 3x3 encontra-se no meio da página).

Para os exercícios abaixo, desligue a folha de estilos usada nos exercícios anteriores (mude o nome ou remova o elemento <LINK>) para que a página fique limpa outra vez. Use uma nova folha de estilos para aplicar as alterações a seguir.
8. Defina classes sec2, sec3, sec31 e sec32 para as seções (<DIV>) do documento Style-Test.html. As seções estão indicadas em comentários HTML (por exemplo: <!--Seção 2 -->). Aplique um fundo diferente (imagem ou cor) nessas seções para diferenciá-las das ou-tras.
9. Tire os sublinhados de todos os links e substitua-os por um fundo cinza claro.
10. Faça com que o link ativo (active) fique em negrito; que o link normal tenha tamanho 10pt e que mostre fundo amarelo quando o mouse estiver sobre ele (hover); e que o link visita-do não tenha mais cor de fundo mas recupere o sublinhado. Obs: Para fazer um link ainda não

visitado, faça um link para qualquer recurso do sistema de arquivos; para ver o link ati-vo, clique no link e segure o mouse.

11. Faça com que:

a) todos os itálicos dentro de negritos sejam colocados em maiúsculas (use text-transform: uppercase).

b) todos os negritos dentro de itálicos sejam sublinhados

c) todos os negritos que estejam dentro de um bloco itálico que está dentro de um bloco LI que está dentro de uma UL que está em outra UL, sejam colocados em fonte arial, em maiúsculas e em vermelho.

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

13. Faça com que os de seus parágrafos sejam 20% maiores que o texto normal destes parágrafos.

14. a) Aplique um espaçamento de 1 cm entre palavras de um parágrafo seu texto (isto poderá não funcionar devido à falta de suporte dos browsers). b) Aplique um espaçamento de 1 cm entre as letras de outro parágrafo. Teste nos dois browsers.

15. Defina todos os títulos H2 como sendo caixa alta (uppercase).

16. Experimente com as propriedades text-decoration (use overline e outras propriedades em blocos criados para mostrar cada propriedade.

17. Elimine o espaçamento entre os parágrafos (<P>) usando {margin-top: 0pt}, endente a primeira linha e coloque todos os seus parágrafos, com exceção dos parágrafos da terceira seção, com alinhamento justificado. O alinhamento deve ser aplicado apenas nos parágrafos e não em listas ou tabelas.

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

19. Inclua a imagem rabbit.gif (ou outra qualquer do subdirectório 3_Imagens do CD do A-SIT) no fundo da página StyleTest.html (usando uma nova folha de estilos: back-ground.css). Experimente com posicionamento, fazendo a imagem ficar fixa em vez de ro-lar na tela. Teste nos dois browsers. Experimente com repetições, fazendo a imagem repe-tir na vertical, na horizontal e não repetir. Veja o que acontece nos dois browsers.

20. Numa outra folha de estilos (para este exercício), posicione a imagem no centro da página, sem repetições e uma outra imagem no centro da tabela, também sem repetições.
21. Descreva o funcionamento das seguintes tags HTML:
- `<H3> ... </H3>`: _____
 - `<HR>`: _____
 - `<Form ...> ... </Form>`: _____
 - `<input type=text ...>`: _____
 - `<input type=hidden ...>`: _____
 - `<select ...> ... </select>`: _____
 - `<textarea ...> ... </textarea>`: _____
22. Construa um formulário HTML com os seguintes campos:
- Nome: Texto, usuário pode digitar no máximo 30 caracteres.
 - Idade: Caixa de seleção (lista ou combo) com os itens:
 - Entre 0 e 18
 - Entre 19 e 25
 - Entre 26 e 35
 - 36 ou Mais
 - Sexo: Dois botões de radio, tendo um radio o value “Masculino” e outro radio com o value “Feminino”.
 - E-Mail: Texto, usuário pode digitar no máximo 80 caracteres.
 - Observações: TextArea com sete linhas e 40 colunas.
23. Implemente um teclado virtual, utilizando HTML e JavaScript.
- Crie um formulário, coloque um `<input type=text>` para armazenar os dados sendo digitados no teclado virtual
 - Implemente o teclado virtual através de botões `<input type=button>` ou através de links `1`.
 - O teclado tem teclas de zero a nove para o usuário poder digitar.
 - Criar uma funcionalidade para apagar o último caractere digitado.
24. Qual a utilidade das funções alert, confirm e prompt?
25. Crie uma função JavaScript para facilitar o processo de abertura de janela, que tenha parâmetros booleanos indicando quais parâmetros da janela queremos habilitados ou não.
26. Crie um formulário com duas caixas de texto, data inicial e data final. Faça um botão para submeter o formulário e no “onValidate” do formulário, coloque código para assegurar que a data inicial e final estão corretas e que a data inicial é menor que a final. Utilize o JavaScript para preencher as “/” automaticamente.
27. Faça uso das funções JavaScript para validar campo vazio, para validar campos obrigatórios de um formulário durante a semana.

28. Como um desafio da disciplina, monte com o que você sabe uma página pessoal para divulgar os seus trabalhos dentro da disciplina e deixando informações que você considera importante.
29. Faça testes modificando os atributos do arquivo .css. Crie blocos personalizados, e associe as classes definidas e faça testes de apresentação destas classes.
30. Faça uma função em JavaScript que receba um parâmetro numérico e crie um Array com onúmero de elementos passado como argumento e o inicie com valores entre 1 e 100.
31. Faça uma função em JavaScript que receba como argumento um array, procure o maior elemento do array e devolva o seu valor.
32. Faça uma função em JavaScript que receba como argumento um array, procure o maior elemento do array e devolva a sua posição no array (índice entre 0 e Dimensão do Array-1).
33. Faça uma função em JavaScript que receba como argumento um array e calcule a média dos elementos do array.
34. Faça uma função em JavaScript que receba como argumento um array e devolva um array com os elementos pela ordem inversa. Faça duas versões da função, uma em que é alterado o array passado como argumento e outra em que o array passado como parâmetro não é alterado.
35. Faça uma função JavaScript que receba como argumento um array e devolva o array ordenado. A função deve alterar o array passado como parâmetro.
36. Faça uma função JavaScript que receba como argumento o dia, mês e ano e devolva um objecto data correspondente.
37. Faça uma função JavaScript que receba uma data como argumento e devolva a data correspondente a uma semana depois no formato (ex: “Domino, 17 de Novembro de 2002”).
38. Faça uma função JavaScript que receba uma data como argumento e devolva o dia do mês.
39. Faça uma função JavaScript que receba uma data como argumento e devolva o mês entre [1,12].
40. Faça uma função JavaScript que receba uma data como argumento e devolva o ano. SSIC – Paulo Araújo – 17/11/2002 2/2
41. Faça uma função JavaScript que receba uma data como argumento e devolva o dia da semana.

42. A tag HTML <INPUT> utilizada para criar elementos de formulários **NÃO** aceita o atributo:
- a) ALT
 - b) NAME
 - c) SIZE
 - d) TYPE
 - e) VALUE
43. Em HTML um programador que deseja adicionar ao código HTML comentários invisíveis ao navegador pode utilizar os delimitadores:
- a) * e *
 - b) // e \\
 - c) <* e *>
 - d) e
 - e)
44. Observe as seguintes afirmativas sobre tags no HTML :
- I - OL - cria listas ordenadas;
 - II - BR - cria uma quebra de linha;
 - III - P - cria novos parágrafos;
 - IV - UL - cria listas não ordenadas.
- Está(ão) correta(s) a(s) afirmativa(s):
- a) **I, somente.**
 - b) **II, somente.**
 - c) **I, II e III, somente.**
 - d) **II, III e IV, somente.**
 - e) **I, II, III e IV.**
45. Crie uma página, em HTML, contendo uma lista enumerada com os nomes de todos os alunos da turma.
46. Crie uma página com as seguintes hiperligações (links):
Página 1
Página 2
Endereço de email
47. Utilizando HTML crie uma tabela, contendo o nome das disciplinas que frequenta, a sua carga horária e o semestre do curso a que pertence.
48. Crie uma página contendo um formulário, contendo um campo de texto, de selecção e entrada de dados.

49. Relativamente ao texto anterior, inclua os atributos que considerar necessários de modo a que os cabeçalhos sejam indentados do seguinte modo:

- h1 indentação normal
- h2 Alinhar pela esquerda
- h3 Alinhar ao centro
- h4 Alinhar pela direita

50. Faça uma folha de estilo que permita obter um texto de cor vermelha no primeiro parágrafo, verde no segundo parágrafo e cinza para o terceiro.