Esta apostila tem por objetivo mostrar de modo fácil como criar websites utilizando as linguagens HTML e CSS; e através do desenvolvimento de exemplos práticos, o leitor conhecerá de forma resumida e direta os principais recursos que envolve o desenvolvimento de layouts web.

Apostila de programação para web: HTML e CSS

Prof. Msc. Regilan Meira Silva
http://www.regilan.com.br
regilan@hotmail.com.

Sobre este material

Esta apostila tem como objetivo auxiliar os estudantes de escolas técnicas e nível superior, na aprendizagem de criação de websites utilizando as linguagens HTML e CSS. Esta apostila não substitui os livros, sua finalidade é criar um roteiro resumido do ensino-aprendizagem realizado em sala de aula.

Este material foi construído a partir de slides elaborados para as minhas aulas no IFBA – Campus Ilhéus e também a partir apostilas, tutoriais, dicas e demais fontes de dados, obtidos a partir de pesquisas em sites de buscas na Internet.

Por fim, este material é distribuído de forma gratuita, sendo vedado qualquer tipo de comercialização.

Sumário

1. INTRODUÇÃO

O sistema com o qual está construído a web se chama hipertexto e é um emaranhado de páginas conectadas com links. A web não só se limita a apresentar textos e links, também pode nos oferecer imagens, vídeos, áudio e todo tipo de apresentações, chegando a ser o serviço mais rico em meios da Internet. Por esta razão, para nos referir ao sistema que implementa o web (hipertexto), foi acunhado um novo termo que é hipermídia, fazendo referência a que o web permite conteúdos multimídia.

A construção de páginas web baseiam-se nas linguagem XHTML + CSS + JAVASCRIPT, que são linguagens utilizadas para estruturar e formatar o conteúdo de um website. Quando vemos uma página web em nosso navegador, ou cliente web, ela parece ser uma só entidade, mas não é assim. Uma página WEB é composta por uma infinidade de diferentes arquivos, como são as imagens, os possíveis vídeos e o mais importante: o código fonte.

O código das páginas está escrito em uma linguagem chamada HTML, que indica basicamente onde colocar cada texto, cada imagem ou cada vídeo e a forma que terão ao serem colocados na página.

Cascading Style Sheets (ou simplesmente CSS) é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML. Seu principal benefício é prover a separação entre o formato e o conteúdo de um documento.Ao invés de colocar a formatação dentro do documento, o desenvolvedor cria um link (ligação) para uma página que contém os estilos, procedendo de forma idêntica para todas as páginas de um portal. Quando quiser alterar a aparência do portal basta portanto modificar apenas um arquivo. Cabem a CSS todas as funções de apresentação de um documento e ao HTML todas as funções de marcação e estruturação de conteúdos. Uma página web é composta então de CSS + HTML.

JavaScript é uma linguagem de programação criada principalmente para as funções de validação de formulários no lado cliente (programa navegador) e interação com a página.

Assim, foi feita como uma linguagem de script. Javascript tem sintaxe semelhante à do Java, mas é totalmente diferente no conceito e no uso. Podemos fazer com JavaScript as seguintes operações:

- Adicionar mensagens que rolam na tela ou alterar as mensagens na linha de status do navegador.
- Validar os conteúdos de um formulário e fazer cálculos.
- Exibir mensagens para o usuário, tanto como parte de um pagina da Web como em caixas de alertas.
- Fazer animações de imagens ou criar imagens que mudam quando você move o mouse sobre elas.
- Detectar o navegador em utilização e exibir conteúdo diferente para navegadores diferentes.
- Detectar plug-ins instalados e notificar o usuário se um plug-ins foi exigido.
- Etc.

Esta apostila tem o objetivo de apresentar os recursos presentes na linguagem HTML, CSS e Javascript para a criação de websites estáticos, sendo assim voltada a parte introdutória dos estudos relacionados ao desenvolvimento para web.

2. PRIMEIROS ELEMENTOS HTML: PARÁGRAFOS, CABEÇALHOS E LISTAS

A finalidade da (X)HTML, desde sua invenção, é a de ser uma linguagem de marcação e estruturação de hipertextos. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (áudio, vídeo, etc.), conectados por webs ou hiperlinks. O HTML fornece a base para a construção de sistemas hipertexto padronizados, consistindo de documentos que aplicam os padrões de maneira particular. Para escrever (X)HTML existem elementos que são representados por tags ou etiquetas.

Tags são pares de sinais destinados a englobar conteúdos e têm finalidade informar ao navegador sobre qual o tipo de conteúdo está nela contido.

Observe a marcação a seguir:

```
<tag inicial> Nonon OnoNonono </tag final>
```

A sintaxe geral da HTML segue o modelo mostrado acima, ou seja, uma marcação indicando o início e outra mostrando o fim de um conteúdo. As marcas inicial e final são chamadas de tags, que, além de delimitar conteúdos, informam a natureza desses conteúdos. Um par de tags constitui um elemento.

Observe a marcação (X)HTML para um parágrafo:

```
<p> Texto de um parágrafo </p>
```

Escrevem-se as tags de abertura e de fechamento entre os sinais: **"<" e ">"** e usa-se uma barra invertida **"\"** logo após o sinal **"<"** para indicar a tag de fechamento.

Existem alguns elementos que são representados por uma tag. São os elementos vazios, assim chamados por não englobarem conteúdos. Tais elementos são empregados na marcação para acrescentar informação ao documento. **Há um elemento (X)HTML destinado a causar uma quebra de linha em um texto, para forçar um quebra de linha existe o elemento "br". Para efeito de padronização o elemento "br" é representado com uma barra antes do sinal de fechamento da tag e o espaço em branco antes da mesma.**

Exemplo tag **"br"**:

```
<p>Texto na primeira linha <br />Texto na segunda linha</p>
```

Atributos são informações adicionais sobre um elemento HTML. Os atributos são definidos dentro da tag de abertura do elemento. No exemplo a seguir, o atributo style é utilizado para um elemento parágrafo.

Exemplo:

```
<p style="color:red"> Parágrafo na cor vermelha </p>
```

A sintaxe para atributos consiste no nome do atributo seguido por um sinal = (igual) e o valor do atributo escrito entre aspas. Outro exemplo:

```
<a href="notas.html"> Notas da IV Unid.</a>
```

A marcação acima diz ao navegador que o texto é um hiperlink que remete o usuário ao documento que contém as notas da IV Unidade. Estudaremos neste capítulo os seguintes elementos:

- Tags para estrutura geral da página:

```
<html>, <head> e <body>
```

- Tags para títulos, cabeçalhos e paragráfos

```
<title>, <h1> a <h6> e <p>
```

- Tags para comentários

```
<!-...->
```

- Tags para lista

```
<ol>, <ul>, <li>, <dt> e <dd>
```

2.1. Tags para estrutura geral de uma página: <html>, <head> e <body>

As tags para estrutura geral de uma página HTML são: <html>, <head> e <body>. A TAG **<html>** engloba toda a marcação HTML do documento, que é chamado de elemento raiz do documento. Os demais códigos HTML estão localizados entre as suas tags de abertura e fechamento.

Exemplo:

```
<html>
... A sua página
</html>
```

O elemento **<head>** Especifica que as linhas dentro dos pontos de início e término da tag são destinadas a agrupar outros elementos contendo informações sobre o documento, como título do documento, folha de estilo(css), links entre outras informações sobre o conteúdo do documento.

Exemplo:

```
<html>
   <head>
      <title>Este é o título. Veja mais nos próximos
slides</title>
   </head>
```

```
</html>
```

O elemento **<body>** identifica o corpo da página. Todo o conteúdo da página deve estar dentro da tag <body>:

```
<html>
   <head>
      <title>Este é o título. Veja mais no próximo
slide.</title>
   </head>
   <body>
     ... A sua página...
   </body>
</html>
```

O elemento **<title>** indica o título da página e deve ser especificado entre os elementos <head> ... </head>. O titulo aparece na barra de títulos do navegador e é utilizado em sistemas de buscas e como favoritos do navegador.

2.2. Tags para cabeçalho e paragrafo: <p>, <h1> a <h6>

Os cabeçalhos **(<h1<, h2>, <h3>, <h4>, <h5>, <h6>)** servem para dividir seções de texto. Exemplo:

```
<h1>Faturamento da Indústria Paulista cai 13,4 % em um ano </h1>
```

Ao contrário dos títulos, os cabeçalhos podem ter qualquer extensão e deverão ser especificados no corpo do documento **<body> </body>.**

Existem seis níveis de cabeçalho, veja exemplo:

```
<body>
<h1>Este é um cabeçalho de nível 1</h1>
<h2>Este é um cabeçalho de nível 2</h2>
<h3>Este é um cabeçalho de nível 3</h3>
<h4>Este é um cabeçalho de nível 4</h4>
<h5>Este é um cabeçalho de nível 5</h5>
<h6>Este é um cabeçalho de nível 6</h6>
</body>
```

O código HTML de cabeçalhos mostrado acima será apresentado assim no navegador:

Este é um cabeçalho de nível 1
Este é um cabeçalho de nível 2
Este é um cabeçalho de nível 3
Este é um cabeçalho de nível 4
Este é um cabeçalho de nível 5
Este é um cabeçalho de nível 6

OBS: Quanto menor o número do cabeçalho, maior será o tamanho da fonte

Para separar blocos de texto em paragráfos, usamos o elemento paragrafo: **<p>**. Exemplo:

```
<body>
   <p> Os parágrafos delimitados por etiquetas “p” podem ser
facilmente justificados à esquerda, ao centro ou à direita,
especificando tal justificação no interior da etiqueta por meio
de um atributo align. Um atributo não é mais do que um parâmetro
incluído no interior da etiqueta que ajuda a definir o
funcionamento da etiqueta de uma forma mais pessoal.
   </p>
   <p>
      Os atributos têm seus valores indicados entre aspas (“).O
atributo align toma determinados valores que são escritos entre
aspas. Em algumas ocasiões necessitamos especificar alguns
atributos para o funcionamento correto da etiqueta. Em outros
casos, o próprio navegador toma um valor definido por padrão.
Para o caso de align, o valor padrão é left.
   </p>
</body>
```

2.3. Tags para listas: <ul>, <ol> a

Existe três tipos de listas:

- Listas numeradas, ou classificadas, geralmente marcadas com números;
- Listas com marcadores, ou listas não classificadas;
- Listas de glossário, nas quais cada item na lista tem um termo e uma definição para ele, e que é organizada de forma que o termo esteja de alguma forma selecionado ou destacado.

As listas numeradas são listas nas quais cada item é numerado sequencialmente. Para a lista numeradas utiliza-se a tag **
...
**. Para os itens lista utiliza-se a tag **
...
**. Exemplo:

```
<body>
   <p> Instalando o seu novo sistema operacional. </p>
   <ol>
      <li> Insira o CD-ROM no seu drive de CD-ROM. </li>
      <li> Selecione EXECUTAR. </li>
      <li> Digite a letra para o drive do seu CD-ROM. </li>
      <li> Siga as instruções do programa de instalação. </li>
      <li> Reinicie o seu computador depois de instalar todos os
           arquivos.</li>
      <li> Cruze os dedos </li>
   </ol>
</body>
```

O código HTML da lista acima será apresentado assim no navegador:

Instalando o seu novo sistema operacional

1. Insira o CD-ROM no seu drive de CD-ROM.
2. Selecione EXECUTAR.
3. Digite a letra para o drive do seu CD-ROM.
4. Siga as instruções do programa de instalação.
5. Reinicie o seu computador depois de instalar todos os arquivos.

6. Cruze os dedos

O atributo start, aplicado a tag <ol> indica o valor inicial da lista que é será sempre um valor numérico.

Exemplo:

```
   <ol start="10">
      <li> Insira o CD-ROM no seu drive de CD-ROM. </li>
      <li> Selecione EXECUTAR. </li>
      <li> Digite a letra para o drive do seu CD-ROM. </li>
      <li> Siga as instruções do programa de instalação. </li>
      <li> Reinicie o seu computador depois de instalar todos os
           arquivos.</li>
      <li> Cruze os dedos </li>
   </ol>
```

As listas não classificadas, são listas nas quais cada item é representado por marcadores (bullets). Para a lista não-classificadas utiliza-se a tag <ul>... e para os itens lista utiliza-se a tag <li>.... Exemplo:

```
<body>
   <p> Coisas que eu gostaria de fazer de manhã. </p>
   <ul>
      <li> Tomar uma xícara de café. </li>
      <li> Ver o sol nascer. </li>
      <li> Ouvir o canto dos pássaros. </li>
      <li> Ouvir o vento assobiando nas árvores. </li>
      <li> Amaldiçoar os barulhos de construção por trazerem o
mau humor.</li>
   </ul>
</body>
```

O código HTML apresentado acima, será apresentado assim no navegador:

Coisas que eu gostaria de fazer de manhã.

- Tomar uma xícara de café.
- Ver o sol nascer.
- Ouvir o canto dos pássaros.
- Ouvir o vento assobiando nas árvores.

- Amaldiçoar os barulhos de construção por trazerem o mau humor.

Uma lista de glossário possue duas partes.

- Um termo;
- A definição do termo.

Para o termo usa-se a tag <dt> e para a definição do termo usa-se a tag <dd>. Toda lista de glossário é indicada entre as tags <dl>...</dl>. Exemplo:

```
<body>
   <dl>
      <dt> Manjericão </dt>
      <dd>
         Anual. Pode crescer até quatro pés de altura, o
         perfume de suas pequenas flores brancas é paradisíaco.
      </dd>
      <dt> Orégano </dt>
      <dd>
         Perene. Espalha raízes no subsolo e é difícil de ser
         evitado quando se estabelece.
      </dd>
   </dl>
</body>
```

O código HTML apresentado acima, será apresentado assim no navegador:

Manjericão

Anual. Pode crescer até quatro pés de altura, o perfume de suas pequenas flores brancas é paradisíaco.

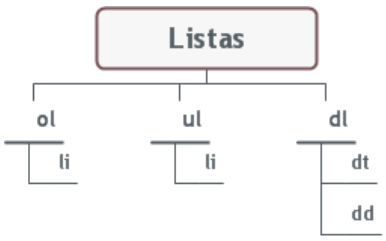
Orégano

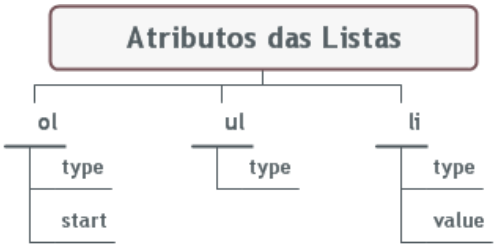
Perene. Espalha raízes no subsolo e é difícil de ser evitado quando se estabelece.

Um resumo dos elementos básicos pode ser representado pelas imagens a seguir:

TAG	Descrição
<html>	Indica o início de um documento HTML.
<head>	Indica um prólogo de um documento HTML.
<body>	Indica o corpo de um documento HTML.
<title>	Indica o título de um documento HTML.
<h1> a <h6>	Cabeçalhos do nível 1 ao nível 6.
<p>	Indica um parágrafo
<!-...->	Indica um comentário

TAG	Descrição
	Indica uma lista ordenada.
	Indica uma lista não ordenada.
	Indica um item da lista.
<dl>	Indica uma lista de definição ou glossário.
<dt>	Indica o termo da lista de definição.
<dd>	Indica a definição do termo da lista de definição.





Nas imagens a seguir são apresentandos algumas páginas web que utilizam as tags apresentadas neste capítulo.

Figura 1. Site globo.com

Figura 2. Site tudogostoso.com.br

Figura 3. Site folha.com.br

2.4. Exercícios práticos

Questão 01. Com base nas TAGS já estudas, desenvolver o código HTML para a estrutura abaixo. **Observações: Utilize lista de definição para os termos e definição das doenças listas**

DICIONÁRIO MÉDICO

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

I (1 - 7):

Icterícia
: Pigmentação amarelada da pele e mucosas devido ao aumento da concentração de bilirrubina no sangue. Pode ser acompanhada de sintomas como colúria (ver), prurido, etc. Associa-se a doenças hepáticas e da vesícula biliar, ou à hemólise (ver).

Imunodeficiência
: Distúrbio do sistema imunológico que se caracteriza por um defeito congênito ou adquirido em um ou vários mecanismos que interferem na defesa normal de um indivíduo perante infecções ou doenças tumorais.

Impetigo
: Infecção da pele e mucosas, produzida por uma bactéria chamada Estreptococo, e caracterizada pela presença de lesões avermelhadas, com formação posterior de bolhas que contém pus e que, ao romper-se, deixam uma crosta cor de mel. Pode ser transmitida por contato entre as pessoas, como em creches.

Imunização

Processo mediante o qual se adquire, de forma natural ou artificial, a capacidade de defender-se perante uma determinada agressão bacteriana, viral ou parasitária. O exemplo mais comum de imunização é a vacinação contra diversas doenças (sarampo, coqueluche, gripe, etc.).

Inconsciência

Distúrbio no estado de alerta, no qual existe uma incapacidade de reconhecer e reagir perante estímulos externos. Pode apresentar-se em tumores, infecções e infartos do sistema nervoso central, assim como também em intoxicações por substâncias endógenas ou exógenas.

Incontinência

Incapacidade de controlar o esvaziamento da bexiga ou do reto. Como resultado produz-se perda de urina ou matéria fecal involuntariamente. As pessoas com incontinência podem apresentar um defeito adquirido ou congênito no mecanismo esfincteriano, ou alguma anormalidade neurológica que as impeça de reconhecer o estado de plenitude da bexiga ou reto e de promover esvaziamento destes quando for conveniente. .

Infarto

Morte de um tecido por irrigação sangüínea insuficiente. O exemplo mais conhecido é o infarto do miocárdio, no qual se produz a obstrução das artérias coronárias com conseqüente lesão irreversível do músculo cardíaco.

<< ***Anterior Próximo*** >>

Questão 02. Em outra página HTML, desenvolver um código HTML para a estrutura abaixo:

LEI Nº 8.112, DE 11 DE DEZEMBRO DE 1990

O PRESIDENTE DA REPÚBLICA;

Faço saber que o Congresso Nacional decreta e eu sanciono a seguinte lei:

TÍTULO II

Do Provimento, Vacância, Remoção, Redistribuição e Substituição

SEÇÃO I - Disposições Gerais

Art. 5. São requisitos básicos para investidura em cargo público:

I. a nacionalidade brasileira;
II. o gozo dos direitos políticos;
III. a quitação com as obrigações militares e eleitorais;
IV. o nível de escolaridade exigido para o exercício do cargo;
V. a idade mínima de dezoito anos;
VI. aptidão física e mental.ível do músculo cardíaco.

Art. 8. São formas de provimento de cargo público:

1. nomeação;
2. promoção;
3. (Revogado pela Lei nº 9.527, de 10/12/97)
4. (Revogado pela Lei nº 9.527, de 10/12/97)
5. readaptação;
6. reversão;
7. aproveitamento;
8. reintegração;
9. recondução.

Art. 36. Remoção é o deslocamento do servidor, a pedido ou de ofício, no âmbito do mesmo quadro, com ou sem mudança de sede.

Parágrafo único. Para fins do disposto neste artigo, entende-se por modalidades de remoção:

I. de ofício, no interesse da Administração; (Inciso acrescentado pela Lei nº 9.527, de 10/12/97)
II. a pedido, a critério da Administração; (Inciso acrescentado pela Lei nº 9.5 27, de 10/12/97)
III. **a pedido, para outra localidade, independentemente do interesse da Administração:**
    a. para acompanhar cônjuge ou companheiro, também servidor público civil ou militar que foi deslocado no interesse da Administração;
    b. por motivo de saúde do servidor, cônjuge, companheiro ou dependente;
    c. em virtude de processo seletivo promovido

Questão 03. Em algumas páginas WEB é comum o desenvolvedor do site colocar um ícone junto ao endereço do site na barra do navegador. Realize uma pesquisa na Internet sobre o tema e construa uma página HTML com um ícone adicionado ao endereço do site.

EXEMPLO: Ver site do GLOBO.COM, UOL, etc.

3. O ELEMENTO <IMG> E NOÇÕES FUNDAMENTAIS DE CSS: FORMATAÇÃO DE FONTE E BACKGROUND

3.1. Imagens: <img>

Para inserir uma imagem em um documento web, utilizamos a tag **<img>**>. O elemento <img> insere imagens que são apresentadas juntamente com os textos presente no documento. Um atributo SRC especificando o caminho físico onde encontra-se a imagem deve estar presente, da seguinte forma:

```
<img src="URL da imagem" />
```

Podemos referenciar imagens do nosso próprio servidor, ou imagens localizadas em outro servidor. As imagens usadas na Web são armazenadas em arquivos com extensão *.gif, *.jpg (ou *.jpeg), *.png.

Outros atributos podem ser definidos sobre a origem, colocação e comportamento da imagem:

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

TITLE: O atributo title permite que seja configurado um texto que será exibido quando o cursor do mouse estiver sobre a imagem.

WIDTH e HEIGHT: Atributos de dimensão (largura e altura respectivamente) da imagem, em pixels. Uma das vantagens de se usar esses atributos é que o browser pode montar mais rapidamente as páginas, por saber de antemão o espaço que deverá ser reservado a elas.

BORDER: Cria uma borda ao redor da imagem. Para retirar a borda, configure com valor 0(zero). Quanto maior o valor, mais larga será a borda.

ALIGN: O atributo align permite alinhar a imagem com um texto.

- LEFT: a figura é desenhada como uma imagem alinhada à esquerda, com o texto fluindo ao seu redor.
- RIGHT: a figura é desenhada como uma imagem alinhada à direita.
- TOP: a parte superior do texto ao redor é alinhada com a parte superior da imagem.
- MIDDLE: a linha de base do texto ao redor é alinhada com a parte central da imagem.
- BOTTOM: A linha de base do texto ao redor é alinhada com a parte inferior da imagem.

3.2. Noções de CSS: regras da sintaxe, formatação de textos e background

3.2.1. Regras da sintaxe CSS

Folhas de estilo em cascata é um mecanismo simples para adicionar estilos (por exemplo: fontes, cores, espaçamentos) aos documentos web. Segundo os idealizadores da WEB, não cabe à HTML fornecer informações ao agente do usuário sobre a apresentação dos elementos. Por exemplo: cores de fontes, tamanho de textos, posicionamento e todo o aspecto visual de um documento não deve ser função da HTML. Cabem às CSS todas as funções de apresentação de um documento. Sendo assim um website é formado por (X)HTML + CSS. Entre as vantagens de se utilizar CSS merece destacar:

- Facilidade de manutenção: definição de estilos que se apliquem a toda página ou website. As mudanças são feitas de forma centralizada
- Novas possibilidades de apresentação visual: muitas funcionalidades permitidas pela CSS não são suportadas pelo HTML
- Diminuição do tempo de download: não é necessário incluir tags de formatação na página de forma que a leitura do código fonte fica mais fácil

Uma regra CSS é uma declaração que segue uma sintaxe própria e que define como será aplicado estilo a um ou mais elementos HTML . Um conjunto de regras CSS formam uma Folha de Estilos. Uma regra CSS, na sua forma mais elementar, compõe-se de três partes: um seletor, uma propriedade e um valor e tem a sintaxe conforme mostrado abaixo:

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

- Seletor: é o elemento HTML identificado por sua tag, ou por uma classe, ou por uma ID, ou etc., e para o qual a regra será válida (por exemplo: <p>, <h1>, <form>, .minhaclasse, etc...);
- Propriedade: é o atributo do elemento HTML ao qual será aplicada a regra (por exemplo: font, color, background, etc...).
- Valor: é a característica específica a ser assumida pela propriedade (por exemplo: letra tipo arial, cor azul, fundo verde, etc...)

Na sintaxe de uma regra CSS, escreve-se o seletor e a seguir a propriedade e valor separados por dois pontos e entre chaves { }.

Quando mais de uma propriedade for definida na regra, deve-se usar ponto-e-vírgula para separá-las.

Veja exemplos:

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

body
{
```

```
color: #000000;
background: #FFFFFF;
font-weight: bold;
}
```

No exemplo abaixo, o seletor é o corpo da página(body), a propriedade é a cor do texto e o valor é a cor azul.

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

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

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

Para maior legibilidade das folhas de estilo, é uma boa prática escrever cada declaração (propriedade e valor) em linhas distintas. Veja exemplo:

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

Essa prática não é obrigatória! A regra abaixo tem o mesmo efeito da regra acima e ambas as sintaxes estão corretas:

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

Uma regra CSS pode ser aplicada a vários seletores ao mesmo tempo. Para isso separe cada seletor com uma vírgula. No exemplo abaixo agrupamos todos os elementos cabeçalho. A cor de todos os cabeçalhos será verde.

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

Nos exemplos já apresentados, usamos as próprias tags HTML como seletor. Seletores CSS não estão restritos às tags, e podem ser diversas entidades da marcação ou a combinação delas. Seletores podem ser atributos do HTML e seus valores, combinações de tags, entre outros.

Em um site com CSS é comum criar um objeto chamado classe e definir as regras CSS independente de uma tag HTML.

As classes podem ser aplicadas a qualquer elemento HTML. Podemos ainda aplicar estilos diferentes para o mesmo tipo de elemento do HTML, usando classes diferentes para cada um deles.

No CSS a sintaxe consiste na combinação do sinal de ponto (.) imediatamente seguido do nome da classe.No código HTML deverá ser criado um atributo para a classe **(class)**

Pode-se ainda usar o nome do elemento HTML para completar a grafia do seletor.

A sintaxe de uma classe é representada da seguinte forma:

```
.nome_da_classe{propriedade:valor;}
```

Ou:

```
elemento_html.nome_da_classe{propriedade:valor;}
```

O objetivo de criar uma classe é aplicar uma regra comum a somente alguns tipos em diferentes seletores. No CSS a sintaxe consiste na combinação do sinal de ponto (.) imediatamente seguido do nome da classe.

No código HTML deverá ser criado um atributo para a classe **(class)**. Pode-se ainda usar o nome do elemento HTML para completar a grafia do seletor. Veja exemplo a seguir:

```
formatacao1
{
color:#0099FF;
border-color:#000000;
border:thin;
}

p.formatacao2
{
color:#99CC33;
border-color:#00FF00;
border:medium;
}
```

No documento HTML as regras são aplicadas conforme abaixo:

```
<p class =“formatacao1"> este é o parágrafo com a primeira formatação.</p>
<p class =“formatacao2"> este parágrafo com a segunda formatação. </p>
```

Outro seletor bastante utilizado é o seletor ID. Este seletor difere do seletor de classe, por ser ÚNICO. Um seletor ID deve ser aplicado a UM e somente UM elemento HTML dentro do documento.

A sintaxe do seletor ID é semelhante a do seletor classe, a diferença é que usamos o sinal de "#" ao invés do "."

Veja a sintaxe abaixo:

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

No documento HTML, aplicamos o seletor ID através do atributo ID, conforme exemplo abaixo:

```
<p id="esporte"> Parágrafos relativo a esportes.</p>

<p id="tecnlogia"> Parágrafos relativo a tecnologia.</p>
```

Você pode inserir comentários nas CSS para explicar seu código, e principalmente ajudá-lo a relembrar de como você estruturou e qual a finalidade de partes importantes do código.

Para adicionar comentários ao código CSS utilizamos a seguinte sintaxe:

```
/* COMENTÁRIOS DE UMA LINHA */

/* COMENTÁRIOS
DE MÚLTIPLAS
LINHAS */
```

3.2.2. Métodos de vinculação do CSS ao HTML

As folhas de estilos são vinculadas no documento HTML de três formas.

- Estilo Externo (linkadas ou importadas);
- Estilo Interno;
- Estilo Inline.

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

Uma folha de estilo externa é ideal para ser aplicada a várias páginas. Com uma folha de estilo externa , podemos alterar a aparência de um site inteiro mudando um arquivo

apenas (o arquivo da folha de estilo). Para incluir um arquivo externo .css a um documento html, usamos a tag <link> entre as tags <head> </head> de acordo com a seguinte sintaxe:

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

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

Uma folha de estilo interna, é ideal para ser aplicada a uma única página, dessa forma podemos alterar a aparência de somente um documento.

No estilo interno as regras são declaradas na seção <head> do documento com a tag de estilo <style> e são aplicadas no documento inteiro.

Veja a sintaxe abaixo:

```
<head>
<style type="text/css">
p{
color:#993366;
}
</style>
</head>
```

Uma folha de estilo é dita inline, quando as regras CSS estão declaradas dentro da tag do elemento HTML. No estilo inline as regras são aplicadas diretamente nas tags html e modifica os atributos de uma tag específica no documento, com isto ele perde muitas das vantagens de folhas de estilo pois mistura o conteúdo com a apresentação. Veja exemplo:

```
<p style="color:#0066CC;font-size:20px">
Regilan Meira
</p>
```

Como as definições de estilo podem ser feita de formas por métodos diferentes, podem existir situações onde os estilos são conflitantes. Neste caso existe uma prevalência nos métodos segundo a regra abaixo:

Prevalência em métodos de especificação:

1. Externo
2. Interno
3. Inline

Prevalência na seleção de métodos:

1. Seletor
2. Class
3. ID

3.2.3. Propriedades de fontes e background

As propriedades para as fontes, definem as características das letras que constituem os textos dentro dos elementos HTML. Elas são representadas de acordo com a relação abaixo:

- font-style: efeito de inclinação; ex.: normal, italic;
- font-variant: transforma letras minúsculas em letras maiúsculas; onde as anteriormente em maiúsculo ficarão um pouco maiores das que estavam em minúsculo; ex.: normal, small-caps;
- font-weight: intensidade (espessura) da fonte; ex.: 100..900,normal, bold, bolder lighter, normal, bold, bolder, lighter.
- font-size: tamanho da fonte; ex.: small, medium, large, smaller, larger, 10, 10%;
- font-family – lista com os nomes da fontes que devem ser utilizadas nos textos por ordem de preferência, começando da mais desejada; ex.: serif, Verdana, Arial, “Times New Roman”;
- font: atalho para especificar as propriedades anteriores em um único local.

Exemplos:

```
h4 { font-style: italic; }
h2 { font-variant: small-caps; }
body { font-weight: bold; }
h1 { font-size: 15pt; }
p  { font-family: serif; }
h6 { font-family: Arial, Verdana, Helvetica; }
h5 { font: italic normal bold 14pt Arial; }
```

A propriedade background define as características do fundo dos elementos HTML. Elas são representadas de acordo com a relação abaixo:

- color : cor do texto;
- background-color : cor de fundo; ex.: cor, transparent;

- background-image : imagem que será utilizada como fundo do elemento; e as propriedades a seguir poderão ser usadas;
- background-attachment: define se a imagem ficará fixa ou se acompanhará a rolagem da área de apresentação; ex.: fixed,scroll;
- background-repeat: como a imagem será repetida se o tamanho dela for menor que a área de apresentação; ex.: no-repeat, repeat, repeat-x, repeat-y;
- background-position: posição inicial da imagem na área de apresentação; ex.: 10% 20%, 100 200, left, center, right, top,center, bottom;
- background: atalho para especificar as propriedades anteriores em um único local.

Exemplo:

```
     a {
     color: #0000FF;
     background-color: #FFFF00;
     }
     body
     {
     background-image: url(fundo.gif);
     background-repeat: no-repeat;
     background-attachment: scroll;
     background-position: center top;
     }
     h3
     {
     background: #0000FF url(fundo.gif) no-repeat scroll center
top;
     }
```

3.3. Exercícios práticos

Questão 01. Criar uma regra CSS para formatação de parágrafo abaixo com as seguintes configurações:

- Cor da fonte Amarela
- Espessura da fonte: 900
- Estilo Itálico
- Tipo de fonte Arial
- Cor de fundo verde

Para a regra acima você deverá criá-la utilizando as três formas vistas em aula:

a. Método externo
b. Método interno
c. Método inline

Questão 02. Em relação às propriedades de background, criar uma classe para cada regra CSS abaixo:

a. Cor de fundo de uma página azul claro
b. Cor de fundo de um parágrafo em verde claro
c. Colocar uma imagem de fundo que permita a repetição em qualquer estilo e que não acompanhe o deslocamento da página
d. Colocar uma imagem de fundo e repeti-la na vertical
e. Colocar uma imagem de fundo e não repeti-la
f. Colocar uma imagem de fundo sem repeti-la e centralizada
g. Formatar o fundo de uma página na cor amarela, uma figura de fundo, sem repetição, centralizada e que acompanhe o deslocamento da página

Ao final crie uma página HTML com vários elementos (cabeçalho ou parágrafos) para testar cada uma das regras acima.

Questão 03. Em relação às propriedades de fonte, criar um id para cada uma das regras CSS abaixo:

a. Tipo de letra Arial e cor vermelha
b. Tipo de letra Times, com tamanho 22 e na cor verde
c. Tipo de letra Comic Sans Serif, com tamanho 20, na cor azul e estilo itálico
d. Formatar a fonte de uma página na cor amarela, em negrito, de tamanho 14, com espessura média e tipo de letra Verdana

Ao final crie uma página HTML com vários elementos (cabeçalho ou parágrafos) para testar cada uma das regras acima.

4. LINKS E TABELAS HTML E PROPRIEDADES CSS PARA FORMATAÇÃO DE BORDAS, ESPAÇAMENTO E MARGENS.

Neste capítulo conheceremos como fazer a ligação de nossas páginas através de links, organizar dados em tabelas e também o model-box do CSS(margem, borda e espaçamentos. Um resumo dos elementos que iremos aprender:

Html:

- Links: <a href="">
- Áudio e Vídeo: <embed />
- Tabelas: <table> <tr> <td> </td> </tr></table>
- Divs:

CSS

- Margin: margem
- Padding: espaçamento interno
- Border: bordas

4.1. Links

Um link é uma conexão para uma página externa ao documento. Pode direcionar o visitante para outra página do mesmo site ou conduzí-lo à uma página externa ao site. Os links têm sua sintaxe geral conforme a seguinte marcação abaixo:

```
<a href="http://www.regilan.com.br"> Site do Professor Regilan </a>
```

A referência ao documento pode ser feita por:

Caminho Absoluto: corresponde ao endereço completo do documento, utilizado quando os documentos estão em servidores diferentes.

Exemplo:

```
<a href = http://maquina/diretorio/pagina.html>
```

Caminho Relativo: quando o documento a ser acessado está no mesmo servidor que a página atual

Exemplo:

```
<a href = "outrapagina.html">
```

Para criar um link com uma página utilizaremos a tag <a> e alguns atributos como href e target

```
  <a href="endereço da página">texto que será o link</a>
```

Ex.:

```
<a href="c:\site\casa.html">Minha casa</a>
```

O atributo HREF determina a localização do arquivo da sua página ou do local da WEB a que o link se referencia.

A URL (Uniforme Resource Locator) é o endereço do link. O texto entre as tags aparecerá sublinhado, indicando que é um link de hipertexto.

O atributo TARGET refere-se a forma como o link será aberto: na mesma página ou em nova página no navegador.

Podemos colocar uma imagem e configurá-la como link para outra página. Para isso usaremos a tag <a> com a tag <img>

Veja exemplo:

```
<html>
<head><title>Instituto Federal da Bahia</title></head>
<body>
<h3>Clique na imagem abaixo para ir ao site do IFBahia
</h3>
<a href="http://www.ifbahia.edu.br">
     <img src="logomarca_ifbahia.png"/>
</a>
<br><br>

</body>
</html>
```

Uma âncora é um ponto de referência ou endereço que será acessado por um link.

Uma âncora é usada dentro do documento para marcar o início de uma seção do documento.

Suponha que seu texto seja muito grande, o que tornaria trabalhoso para quem estiver vendo a página conseguir se movimentar entre um tópico e outro. Então faça o seguinte: Nomeie uma parte da sua página através da tag : <a name="">

```
<a name="nome"> Conteúdo a ser mostrado: texto, imagem,
etc...
</a>
```

Depois disso escreva a parte da página que você quer associar a esse Nome. Agora crie um link para chegar até esta parte, usando a tag :

```
<a href="#nome">clique aqui</a>
```

Ao clicar na mensagem "Clique Aqui" de sua página , o usuário irá até a parte que você nomeou.

O sinal de cerquilha/tralha (#) é necessário para a âncora, pois avisa ao browser para procurar o link no documento atual !!!

4.2. Vídeo e áudio

Existem vários modos de colocar um vídeo ou arquivo de áudio em seu site. A inserção de vídeo depende bastante do tipo de arquivo de vídeo que temos para inserir em uma página e se o plugin daquele formato está instalado no navegador.

Uma forma simples de adicionar Áudio e Vídeo é utilizando a tag <embed>

```
<embed src="nome do arquivo.xxx">
```

O HTML5 introduz o suporte de mídia embutido por meio dos elementos <audio> e <video>, oferecendo a possibilidade de incorporar facilmente mídia em documentos HTML.

Estes elementos são independentes da instalação de plug-ins e desde que o navegador utilizado suporte HTML os arquivos de áudio e vídeo serão carregados sem a necessidade de instalação de plug-ins.

Para adicionar áudio ou vídeo utilizamos respectivamente as tags:

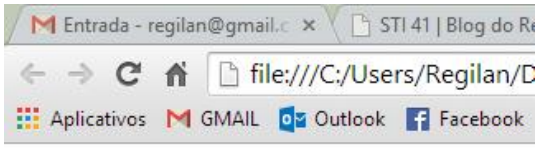
```
<audio src="arquivo de áudio.xxx"> </audio>
<video src="arquivo de vídeo"> </video>
```

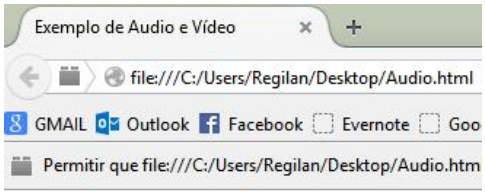
Estes elementos possuem os seguintes atributos:

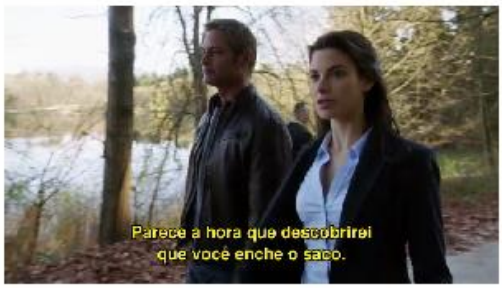
- controls: Mostra os controles padrão para o áudio/vídeo na página.
- autoplay : Faz com que o áudio/vídeo reproduza automaticamente.
- loop : Faz com que o áudio/vídeo repita automaticamente.
- width e height: largura e altura do elemento

Exemplo:

```
<video src="Inteligence Ep01.mp4" controls="controls" autoplay="autoplay"width="300px" height="300px"/>
```

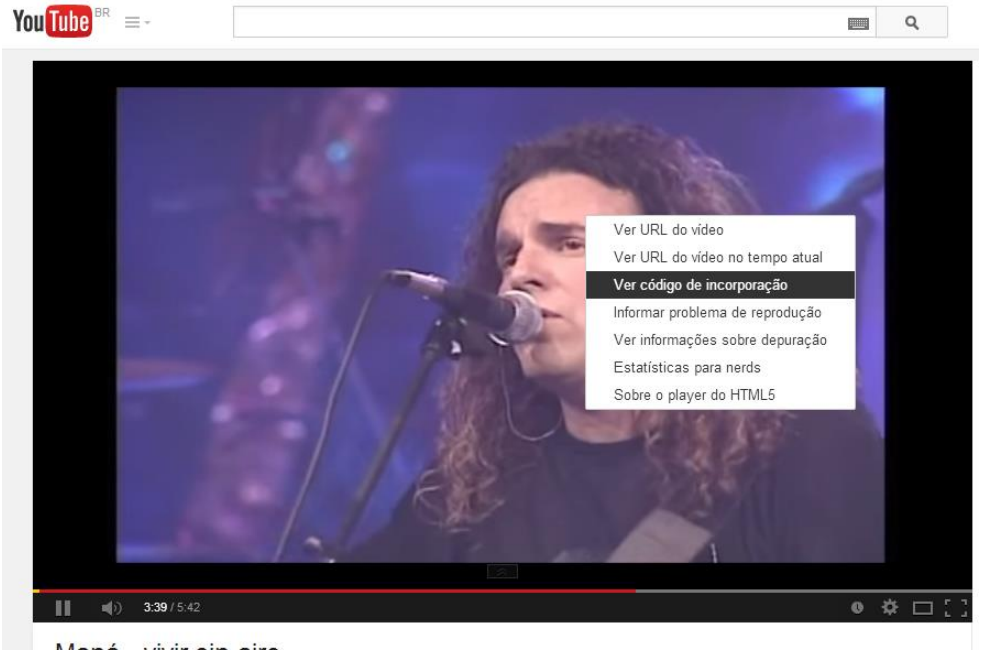






Alguns sites como por exemplo o YouTube disponibiliza a incorporação de vídeos. No caso do YouTube devemos exibir o código de incorporação para incluí-lo em nossa página HTML.

Para isto basta clicar com o botão direito sobre o vídeo e depois em Ver código de incorporação. Em seguida copiamos o código HTML exibido e colamos em nossa página HTML.



4.3. Vídeo e áudio

Tabelas são utilizadas para apresentação de dados em forma de linhas e colunas.

O elemento para se inserir uma tabela é <table>; para iniciar uma linha devemos introduzir a tag <tr> e para uma célula (alguns preferem dizer coluna) <td>.

Todos estes comandos são encerrados como </table> , </tr> e </td> respectivamente.

Existe ainda o comando <th> </th> que permite criar uma coluna de título(cabeçalho da tabela).

```
<table>
<tr>
     <td> </td>
</tr>
</table>
```

Estas tags definem que o conteúdo que está entre elas deve ser organizado na forma de tabela. Para a formatação da tabela podem ser colocados junto da tag <TABLE> os seguintes atributos abaixo:

- align=(left/center/right): alinhamento da tabela em relação ao documento.
- bgcolor: define uma cor para o segundo plano da tabela.
- border=n: coloca uma borda com espessura n (padrão: sem borda).
- cellspacing=n: espaçamento entre as células
- cellpading=n: espaçamento entre a borda de uma célula e seu conteúdo (padrão: 1).
- cols=n: número de colunas de uma tabela. permite o carregamento mais rápido de tabela muito grandes.
- width=n/n%: largura da tabela.

Para criar uma linha em uma tabela, utilizamos a tag: <tr> ... </tr>. A abertura de uma linha da tabela, possui as seguintes opções:

- align=(left/center/right): alinhamento horizontal do conteúdo, tornando-se padrão para toda linha.
- valign=(top/middle/bottom/baseline): alinhamento vertical do conteúdo, tornando-se padrão para toda linha.
- bgcolor: define uma cor para o segundo plano para a linha.

```
<tr> </tr>
```

Uma tabela é formadas por linhas que são representadas pela tag <tr> e por células (colunas) que são representadas pelas tags <th> ou <td>

A tag <th> ... <th> cria uma célula/coluna de título.

A tag <td> ... <td> é utilizada para o conteúdo de uma célula.

As tags <th> e <td> também possuem os mesmos atributos da tag table: align,border,width, bgcolor, etc.

```
<td> </td>
<th> </th>
```

Exemplo de código de criação de tabela com 2 linhas e 3 colunas:

```
<table border=1>
<tr>
<td>primeira coluna </td>
<td>segunda coluna </td>
<td>terceira coluna </td>
</tr>
<tr>
<td> primeira coluna</td>
<td>segunda coluna </td>
<td>terceira coluna </td>
<tr>
</table>
```

primeira coluna	segunda coluna	terceira coluna
primeira coluna	segunda coluna	terceira coluna

É possível englobar colunas e linhas, através dos atributos colspan (para colunas) e rowspan (para linhas). Estes atributos são muito importantes pois nos possibilitam remodelar a disposição das células dentro da tabela.

Veja exemplo:

```
<table border=2 cellpadding=2>
<tr>
<td colspan=2>exemplo do uso do colspan
</td>
```

```
<tr>
<td>célula 1</td> <td>célula 2
</td>
</tr>
</table>
```

exemplo do uso do colspan	
célula 1	célula 2

Apesar dos atributos HTML, presente nos elementos para criação de tabelas, estarem disponíveis para formatação de cores, bordas, espaçamento, largura, altura e outros, a recomendação é que **TODAS AS FORMATAÇÕES POSSÍVEIS SEJAM REALIZADAS ATRAVÉS DO CSS.**

Veremos ainda neste capítulo os elemtnos CSS para formatação de bordas, margens e espaçamentos e que podem ser aplicadas a formatação de tabelas.

4.4. O elemento DIV

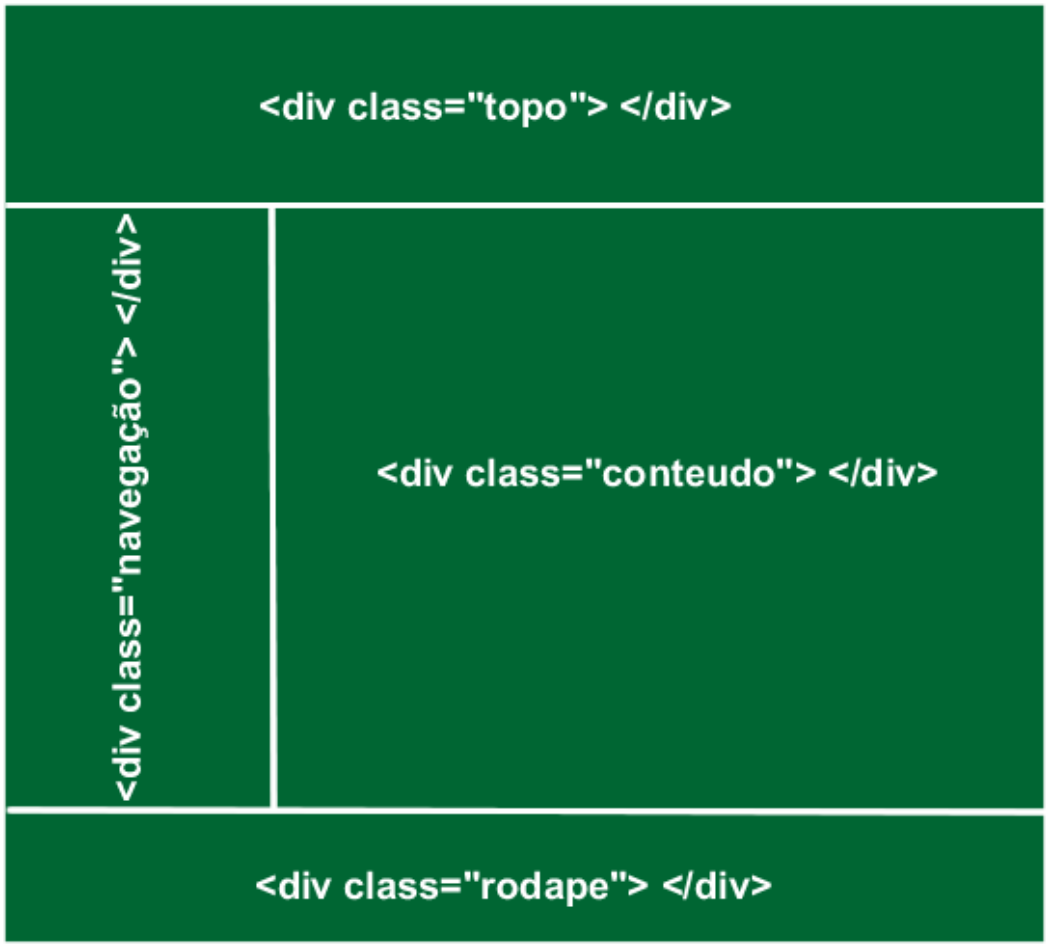
A tag DIV define uma divisão ou um seção em um documento HTML.

DIV na verdade não causa nenhuma diferença visual no código. Ele é considerado um "container", ou seja, uma espécie de "caixa" não visual que você pode, através de script, alterar o conteúdo dele, alterando o código HTML dinamicamente. Ou então é usado para aplicar um estilo (css) em todo o bloco HTML contido dentro do

```
<div id="esporte"> </div>
<div id="economia> </div>
```

Os sites atuais estão sendo produzidos, com a utilização de uma coleção de mais e mais elementos div, em substituição as antigas tabelas que eram usadas para criar o layout de um site. Para cada div é criada uma classe ou id no código css que contém o estilo de uma determinada div no documento HTML. Nos próximos capítulos estudaremos com criar layouts com DIVs e regras CSS.

Exemplo:



4.5. O box-model CSS: margem, borda e espaçamento

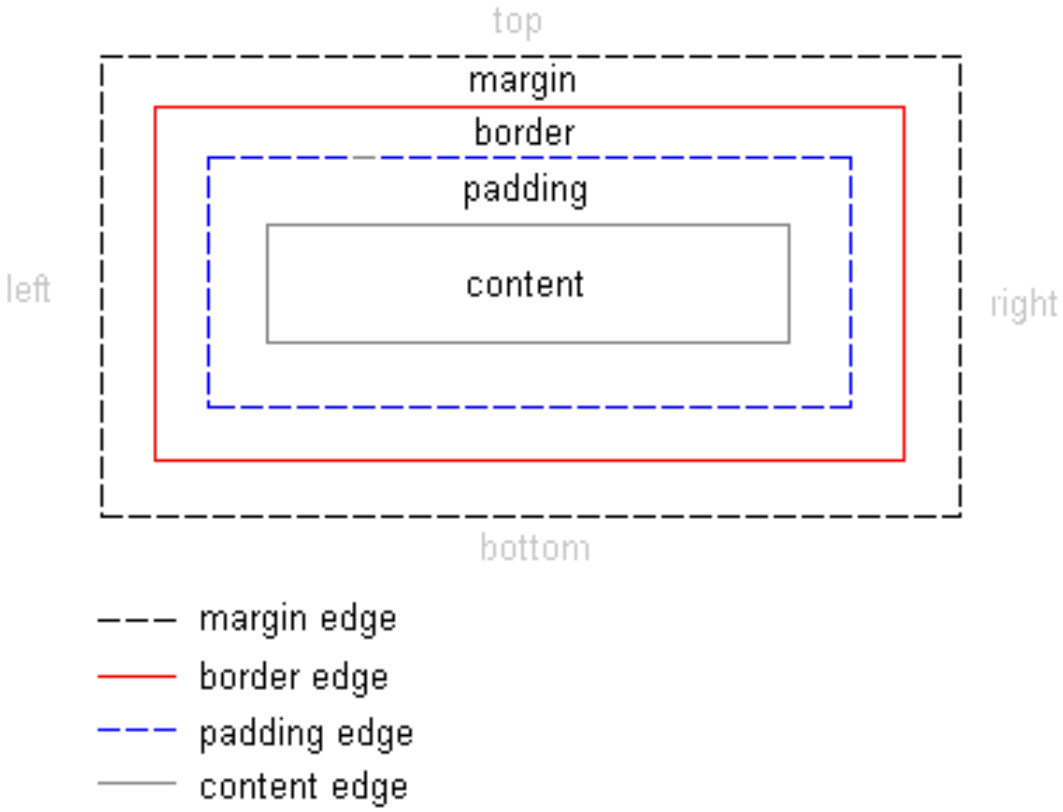
Todo elemento HTML é uma "caixa" retangular a ser apresentada na tela com as estilizações determinadas pelas regras CSS. As caixas são empilhadas uma após a outra e constituídas de margens, bordas, espaçamentos e o conteúdo propriamente dito.

O box model é a especificação que define como o elemento e os seus atributos se relacionam entre si. Podemos dizer que o box model diz aos browsers que uma box tem de largura 100 pixels e de altura 50 pixels, portanto terá de ser apresentada de acordo com estas especificações.

Desta forma, cabeçalhos (<h1>,<h2>,<h3>..), parágrafos (<p>), listas (<ul>, <ol>), formulários (<form>), divisões ( ), e em fim qualquer elemento HTML (tag) é representado por uma caixa. Podemos então dizer que a caixa assim como foi definida é a unidade básica de formatação CSS.

As caixas podem conter ou estar contidas dentro de outras caixas. As caixas CSS são constituídas por 04 (quatro) áreas retangulares, que se desenvolvem de dentro para fora, na ordem listada abaixo:

- conteúdo;
- espaçamentos (padding);
- bordas (border);
- margens (margin).



Neste momento, vale ressaltar sobre as unidades de medidas que podem ser relativas ou absolutas.

Unidades absolutas são unidades de medida de comprimento definidas nos sistemas de medidas pela física , a citar centímetros, milímetros, polegadas, etc. São indicadas para serem usadas quando as mídias de exibição são conhecidas. Como um navegador apresenta um documento de acordo com as configurações do monitor do usuário, este tipo de medida é inviável já que o desenvolvedor não tem como saber antecipadamente ou controlar as configurações.

Exemplos de medidas absolutas são:

- in = polegada
- cm = centímetro
- mm = milímetro
- pt = ponto

- div { margin: 2.5cm; }
- h4 { margin: 2mm; }
- p { font-size: 10pt; }
- .classe { padding: 3pt; }
- hr { width: 14pt; }
- h1 { margin: 1in; }

As unidades de medida relativas são aquelas que tomam como base um valor de referência e são mais apropriadas para ajustes de uso em diferentes tipos de mídia. As quatro unidades de medida relativa são:

- em = 1em é igual ao tamanho de fonte definido para o elemento em questão
- ex = 1ex é igual a altura da letra xis minúscula(x) da fonte definida
- px = ao dispositivo (midia) de exibição;
- %: ... a uma medida previamente definida.

Exemplos:

- div { margin: 1.5em; }
- h4 { margin: 2ex; }
- p { font-size: 14px; }
- .classe { padding: 90%; }
- h1 { line-height: 1.2em }
- p { font-size: 10px }

Voltando ao modelo de caixa(box-model), uma box é constituída por partes distintas:

- A margin é invisível, não possui cor de fundo e não esconde elementos.
- A border permite visualizar os limites visíveis da box.
- O padding define o espaço entre o conteúdo e o border.

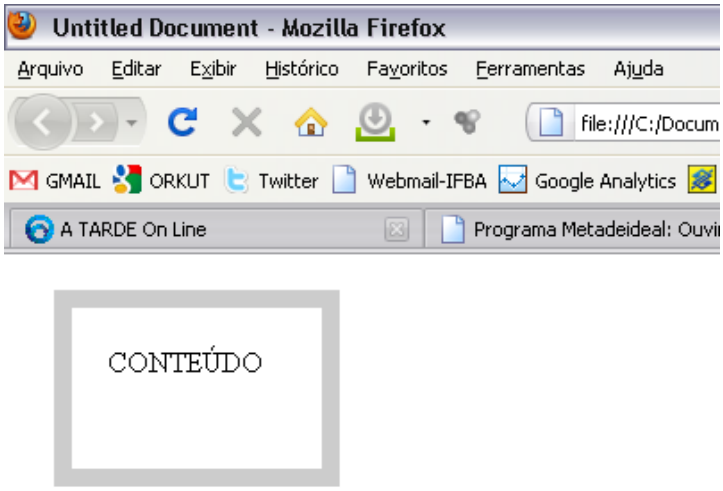
A única parte visível da box apresentada na imagem seria o border, portanto toda a zona a tracejado não estaria visível. Veja exemplo do código CSS de uma classe chamada .box:

```
.box
{
width: 100px;
```

```
    height: 50px;
    border: 10px solid #ccc;
    padding: 20px;
    margin: 20px;
    }
```

Agora o código HTML:

```
    <p class="box">CONTEÚDO</p>
```



A PROPRIEDADE PARA MARGENS, define um valor para espessura das margens dos elementos HTML.

As propriedades para margens são as listadas abaixo:

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

Os valores possíveis para as propriedades das margens são:

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

Exemplos:

```
.box
{
margin-top: 20px;
margin-right: 30mm;
margin-bottom: 5pt;
margin-left: 3in;
}
```

São válidos valores negativos para margem, com o objetivo de sobrepor um elemento. É possível também declarar de forma única a ordem das margens: superior, direita, inferior e esquerda.

A PROPRIEDADE MARGIN admite a sintaxe abreviada, a qual consiste em declarar uma lista de valores separados por um espaço, conforme mostrado.

Há quatro modos de se declarar abreviadamente as margens:

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

Exemplos da propriedade margin com a sintaxe abreviada:

```
.box
{
margin: 20px /* margem de 20px nos quatro lados */
margin: 15px 10px /* margem superior e inferior de 15px e
direita e esquerda de 10px */
margin: 15px 5px 10px /* margem superior 5px, direita e
esquerda 5px e inferior de 10px */
margin: 15px 5px 10px 20px /* margem superior 5px, direita
5px, inferior 10px e esquerda de 20px */
}
```

A PROPRIEDADE PARA ESPAÇAMENTOS, define um valor para os espaçamentos entre o conteúdo e as bordas dos elementos HTML.

As propriedades para margens são as listadas abaixo:

- padding-top: define a espaçamento superior;

- padding-right: define a espaçamento direita;
- padding-bottom: define a espaçamento inferior;
- padding-left: define a espaçamento esquerda;
- padding: maneira abreviada para todas os espaçamentos

Os valores possíveis para as propriedades dos espaçamentos são:

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

Exemplos:

```
.box
{
padding-top: 20px;
padding-right: 30px;
padding-bottom: 5px;
padding-left: 10px;
}
```

A propriedade padding também admite a sintaxe abreviada, a qual consiste em declarar uma lista de valores separados por um espaço. Há quatro modos de se declarar abreviadamente os espaçamentos:

- padding: valor1 - os 4 espaçamentos terão valor1;
- padding : valor1 , valor2 - espaçamento superior e inferior terão valor1 - espaçamento direita e esquerda terão valor2
- padding : valor1 , valor2 , valor3 - espaçamento superior terá valor1 - espaçamento direita e esquerda terão valor2 - espaçamento inferior terá valor3
- padding : valor1 , valor2 , valor3 , valor4 - espaçamento superior, direita, inferior e esquerda nesta ordem.

Exemplos da propriedade padding com a sintaxe abreviada:

```
.box
{
padding: 20px /* espaçamento de 20px nos quatro lados */
padding : 15px 10px /* espaçamento superior e inferior de 15px e direita e esquerda de 10px */
padding : 15px 5px 10px /* espaçamento superior 5px, direita e esquerda 5px e inferior de 10px */
```

```
     padding : 15px 5px 10px 20px /* espaçamento superior 5px,
direita 5px, inferior 10px e esquerda de 20px */
     }
```

A PROPRIEDADE BORDER define a espessura, a cor e o estilo das bordas do box. As propriedades para as bordas são as listadas abaixo:

- border-width: espessura da borda
- border-style: estilo da borda
- border-color: cor da borda

Cada uma destas três características da borda pode ser declarada separadamente para cada lado do box, conforme o slide a seguir:

Borda superior:

- border-top-width: espessura da borda superior
- border-top-style: estilo da borda superior
- border-top-color: cor da borda superior

Borda direita:

- border-right-width: espessura da borda direita
- border-right-style: estilo da borda direita
- border-right-color: cor da borda direita

Borda inferior:

- border-bottom-width: espessura da borda inferior
- border-bottom-style: estilo da borda inferior
- border-bottom-color: cor da borda inferior

Borda esquerda:

- border-left-width: espessura da borda esquerda
- border-left-style: estilo da borda esquerda
- border-left-color: cor da borda esquerda

Podemos ainda abreviar a propriedade border da seguinte forma:

- border-top: maneira abreviada para todas as propriedades da borda superior
- border-right: maneira abreviada para todas as propriedades da borda direita
- border-bottom: maneira abreviada para todas as propriedades da borda inferior
- border-left: maneira abreviada para todas as propriedades da borda esquerda
- border: maneira abreviada para todas as quatro bordas

Os valores possíveis para a propriedade border são:

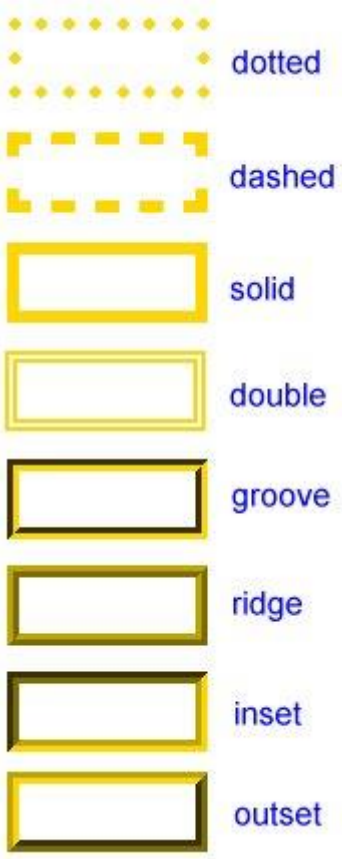
color:

- código hexadecimal: #FFFFFF
- código rgb: rgb(255,235,0)
- nome da cor: red, blue, green...etc

width:

- thin: borda fina
- medium: borda média
- thick: borda grossa
- length: uma medida reconhecida pelas CSS (px, pt, em, cm, ...)

Podemos aplicar oito estilos(style) para bordas ou declarar o valor none para definir a ausência de bordas. O estilo de cada uma é apresentado na imagem abaixo.



Exemplos da propriedade border:

```
p {
border-style: solid;
border-bottom-style: dashed;
border-color:#00FF33
border-top-color:#FF00BB
border-bottom-width: 10px;
border-top-width: 0px; border-right-width: 0px;
border-left-width: 0px;
}
```

Podemos declarar todas as três propriedades das bordas em uma regra única. A sintaxe geral é:

```
border: size style color
```

Exemplo:

```
p { border: thick groove #FF0033 }
```

A seguir será apresentado uma série de efeitos em cabeçalhos que podem ser realizados através da configuração de bordas.

BORDA SIMPLES:

```
h1
{
border-bottom-style:solid;
border-bottom-width:1px;
border-bottom-color:#009966;
}
```

BORDA DUPLA:

```
h1
{
border-bottom-style:solid;
border-bottom-width:1px;
border-bottom-color:#009966;
```

```
    border-top-style:solid;
    border-top-width:1px;
    border-top-color:#009966;
    }
```

Exemplo com bordas simples

BORDA DUPLA E FUNDO:

```
    h1
    {
    border-bottom-style:solid;
    border-bottom-width:1px;
    border-bottom-color:#009966;
    border-top-style:solid;
    border-top-width:1px;
    border-top-color:#009966;
    }
```

Exemplo com bordas dupla e fundo

BORDAS COM ESPESSURAS DIFERENTES E FUNDO:

```
    h1
    {
    background-color:#CCCCCC;
    border-bottom-style:solid;
    border-bottom-width:2px;
    border-bottom-color:#009966;
    border-left-style:solid;
    border-left-width:10px;
    border-left-color:#009966;
    }
```

Bordas com espessuras diferentes e fundo

BORDAS COM ESTILOS DIFERENTES E FUNDO:

```
h1
{
background-color:#CCCCCC;
border-bottom-style:solid;
border-bottom-width:2px;
border-bottom-color:#009966;
border-top-style:dotted;
border-top-width:2px;
border-top-color:#009966;
}
```

BORDAS COM IMAGEM DE FUNDO:

```
h3
{
border-bottom-style:solid;
border-bottom-width:2px;
border-bottom-color:#999999;
padding-left:25px;
background-image:url(square_arrow.gif);
background-repeat:no-repeat;
}
```

BORDAS PERSONALIZADAS:

```
h2
{
background-position:left bottom;
background-image:url(estrela.png);
background-repeat:repeat-x;
padding-bottom:15px;
}
```

Bordas personalizadas

A propriedade border-radius disponibilizada na versão CSS 3 permite criar bordas arredondadas.

Como exemplo usaremos uma DIV e configuramos o código Css para apresentação da borda.

```
#bordasArredondadas
{
width:400px;
height:150px;
background-color:#09F;
border-radius:10px;
}
```

```
<div id="bordasArredondadas">
```

Assim como as propriedades margin, padding e border, é possível definir valores diferentes para cada canto:

border-radius-topleft: para o canto superior esquerdo

border-radius-topright: para o canto superior direito

border-radius-bottomright: para o canto inferior direito

border-radius-bottomleft: para o canto inferior esquerdo

As bordas arredondadas não ficam restritas ao elemento DIV, elas podem ser usadas em botões, campos de formulário, parágrafos, caixas de destaque em uma página de textos, entre outros.

As propriedades estudadas também podem ser empregas para estilização de tabelas:

```
table
{
border-style:solid;
border-color:#0066FF;
border-width:2px;
}
tr td,tr th
{
border-style:solid;
border-color:#0099FF;
```

```
    border-width:1px;
    }
```

Unidades de Medidas

Unidade	Símbolo
Metro	m
Quilo	kg
Litro	l
Joule	j
Watt	w

4.6. Exercícios práticos

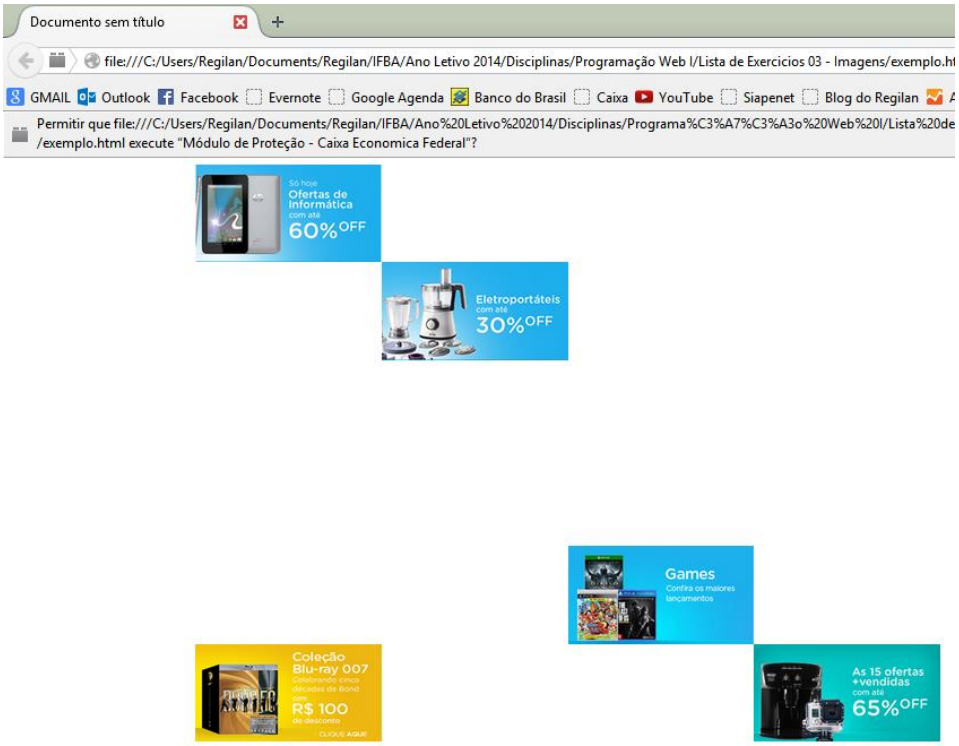
Questão 01. Criar uma regra CSS para formatação de parágrafo com as seguintes configurações:

- Largura do parágrafo: 150px;
- Altura do parágrafo: 100px;
- Cor da fonte branca
- Tipo de fonte Arial
- Cor de fundo verde
- Borda Superior: Espessura de 2px, na cor azul e estilo dashed.
- Borda Esquerda: Espessura de 2px, na cor amarela e estilo dotted.
- Borda Inferior: Espessura de 2px, na cor vermelha e estilo double.
- Borda Direita: Sem borda
- Espaçamento Superior e Esquerda: 10px

Ao final adicione um texto ao parágrafo e verifique se o resultado apresentado está de acordo com a formatação sugerida.

Questão 02. Em relação às propriedades de margens, criar regras CSS de forma que o resultado a ser apresentado seja semelhante a imagem a seguir:

OBS: Todas as imagens utilizadas na página acima estão em uma pasta compactada junto com a atividade.



Questão 03. Utilizando Divs/Tabelas e regras CSS, cria uma página HTML para apresentar um menu conforme a imagem abaixo:

Configure as propriedades: *border* e *padding* de forma que o menu seja apresentado semelhante a imagem lateral.

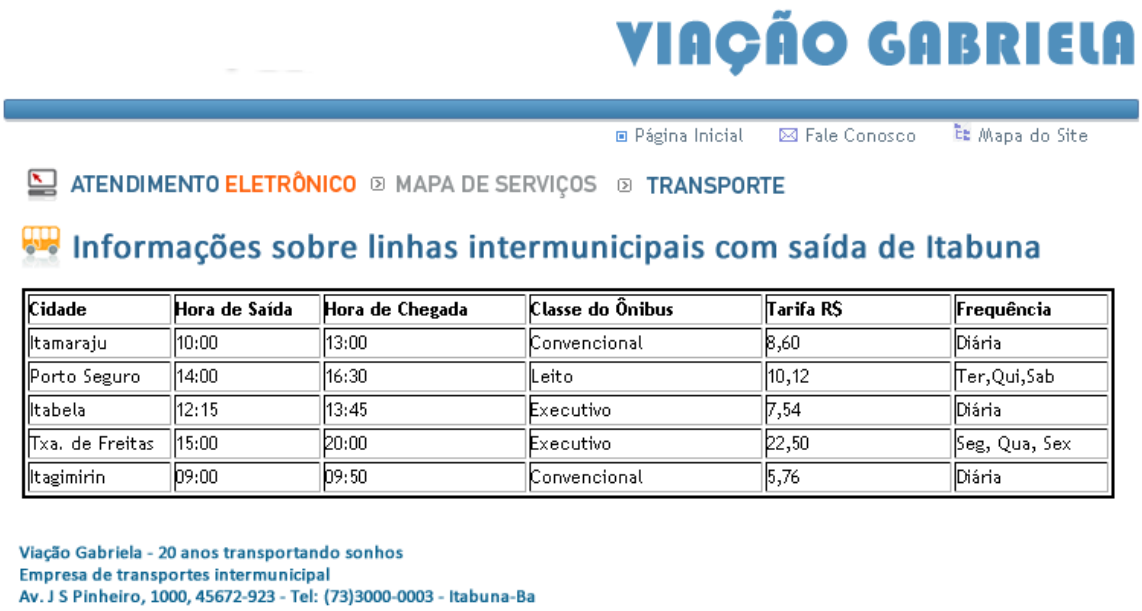
Altere também as propriedades de fonte e cor para os títulos do menu e configure a cor da borda a direita da mesma cor usada para os títulos do menu.

Adicione também links externos ou internos (OBS: De livre escolha) para cada um dos itens do menu.

Questão 04. Utilizando tabelas e regras CSS, crie uma página HTML que apresente o resultado mostrado na imagem a seguir:

CLASSIFICAÇÃO		PG	J	V	SG
1°	Cruzeiro	39	17	12	21
2°	São Paulo	32	17	9	9
3°	Internacional	31	17	9	9
4°	Corinthians	31	17	8	12

Questão 05. Utilizando tabelas e regras CSS, crie uma página HTML que apresente o resultado mostrado na imagem a seguir:



Cidade	Hora de Saída	Hora de Chegada	Classe do Ônibus	Tarifa R$	Frequência
Itamaraju	10:00	13:00	Convencional	8,60	Diária
Porto Seguro	14:00	16:30	Leito	10,12	Ter,Qui,Sab
Itabela	12:15	13:45	Executivo	7,54	Diária
Txa. de Freitas	15:00	20:00	Executivo	22,50	Seg, Qua, Sex
Itagimirin	09:00	09:50	Convencional	5,76	Diária

Viação Gabriela - 20 anos transportando sonhos
Empresa de transportes intermunicipal
Av. J S Pinheiro, 1000, 45672-923 - Tel: (73)3000-0003 - Itabuna-Ba

5. ESTILIZAÇÃO DE TEXTOS, LISTAS E PSEUDO-ELEMENTOS

5.1. Estilização de textos

As propriedades para textos, definem as características dos textos inseridos dentro dos elementos HTML:

- color: cor do texto;
- text-align: alinhamento do texto;
- text-decoration: decoração do texto;
- text-indent: recuo do texto;
- text-transform: forma das letras;
- direction: direção do texto;
- letter-spacing: espaçamento entre letras;
- word-spacing: espaçamento entre palavras;

Em relação ao alinhamento, temos a propriedade text-align que controla o posicionamento horizontal do conteúdo de um elemento nível de bloco. Os valores possíveis são:

- left
- right,
- Center
- justify
- inherit(um valor herdado).

Esta propriedade coloca os textos às esquerda, direita, centralizado ou justificados em relação ao container onde está localizado. No exemplo abaixo, foi criado quatro classes para alterar o alinhamento do texto através da propriedade text-align:

```
.centro
{text-align:center;}

.direita
{text-align:right;}

.esquerdo
{text-align:left;}

.justificado
{text-align:justify;}
```

Outra propriedade de alinhamento é a text-indent que define uma endentação para primeira linha do texto contida em um bloco. Para configurá-lo, adicionamos um medida CSS de comprimento,porcentagem ou um valor herdado.

CSS

```
.exemplo{text-indent:100px;}
```

HTML:

```
<p class="exemplo">Onononono</p>
```

Além de controlar espaçamentos e alinhar textos, podemos utilizar as propriedades CSS para adicionar uma série de efeitos em textos. A primeira propriedade a ser estudada é a propriedade text-decoration que causa um efeito decorativo no texto. Os valores possíveis desse efeito são:

- none(sem decoração)
- Underline
- Overline
- line-through
- blink.

No CSS:

```
.linhacortando{text-decoration:line-through;}
.duaslinhas{text-decoration: overline underline;}
.piscar{text-decoration:blink;}
```

No HTML:

```
<p class="linhacortando">Linha cortando</p>
<p class="duaslinhas">Duas linhas</p>
<p class="piscar">Piscar</p>
```

Resultado no Navegador:

~~Linha cortando~~

Duas linhas

Piscar

OBS: poderá ser configurado um valor negativo para esta propriedade.

Outro efeito em texto é a propriedade ***text-transform*** que controle os efeitos de capitalização dos textos. Os valores possíveis dessa propriedade são:

- none;
- capitalize(primeiro caractere em caixa alta);
- uppercase(todos os caracteres em caixa alta);
- lowercase(caracteres em caixa baixa)
- inherit

Exemplo:

```
p{text-transform:capitalize;}
```

A propriedade text-shadow é utilizado para aplicar uma sombra a um texto.

A propriedade text-shadow é usada da seguinte maneira:

```
 text-shadow: 2px 2px 2px #00f;
```

Os dois primeiros valores são as coordenadas top left, para onde a sombra será projetada. O terceiro valor é o efeito blur da sombra e q quarto valor é a cor da sombra.

Por último temos ainda a propriedade direction controla a direção do texto e apresenta os seguintes valores:

- ltr: texto escrito da esquerda para a direita
- rtl: texto escrito da direita para a esquerda

Exemplo:

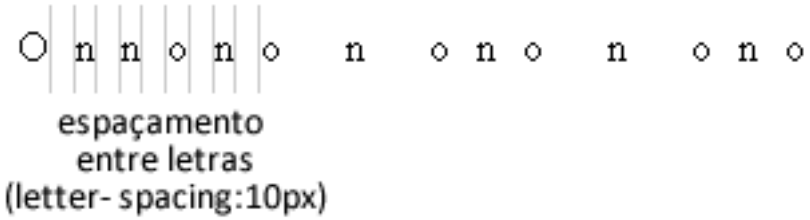
```
p{
direction:rtl;
}
```

As regras CSS dispõem de mecanismos para controle de espaçamentos em fontes e textos, seja na vertical quanto na horizontal. Para controlar o espaçamento entre letras e parágrafos de um texto utilizamos as propriedades:

- letter-spacing;
- word-spacing;
- line-height

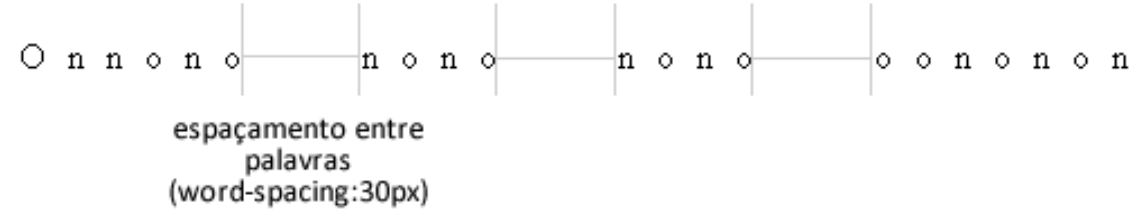
A propriedade letter-spacing controla o espaçamento entre letras de um texto. Veja exemplo:

```
p
{
  letter-spacing:10px;
}
```



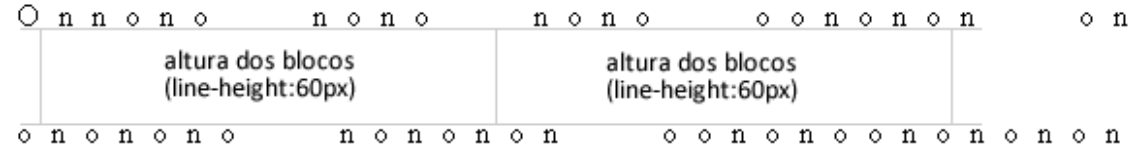
A propriedade word-spacing controla o espaçamento entre as palavras de um texto.

```
p
{
   word-spacing:30px;
}
```



A propriedade line-height controla a altura dos blocos contidos dentro de um elemento nível de bloco.

```
p
{
       line-height:30px;
}
```



5.2. Estilização de listas

HTML aceita listas que podem ser numeradas e não numeradas, sendo possível, inclusive, incluir sub-itens nas listas.

As listas não numeradas são declaradas no HTML pelas tags <ul>

Listas numeradas utilizam-se as tags <ol>

Além da utilização para marcar listas de informações, este elemento é utilizado também para a criação de menus CSS, que estudaremos em próximos capítulos

Para alterar a aparência do marcador de uma lista, usamos a propriedade: list-style-type. Podemos alterar para três tipos: glifos, sistema númerico, sistem alfabético.

Exemplo:

```
ul
{
list-style-type:circle;
}
```

- nononon
- nononon
- nononon

OBSERVAÇÃO: Podemos retirar os marcadores padrão de lista com o uso do valor none, como mostrado abaixo:

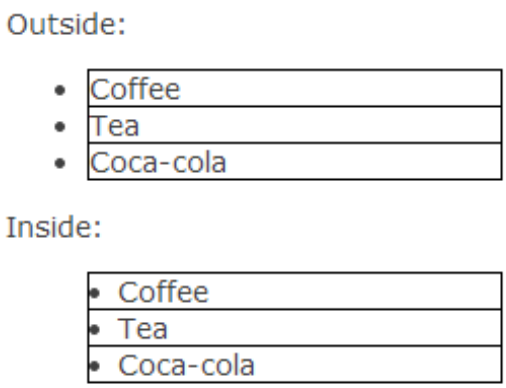
```
ol, ul{list-style-type:circle;}
```

Além dos tipos padrões de marcador de uma lista, podemos definir uma pequena imagem como marcador de uma lista. Essa propriedade substitui automaticamente o marcador padrão por uma imagem construída.

Veja exemplo da regra css abaixo:

```
ul
{
list-style-type:none;
list-style-image:url(target.png);
font-weight:bold;
color:#FF0000;
}
```

Por último, existe a propriedade list-style-position que define a posição do box que contém o marcador da lista em relação ao box principal do elemento li. Os valores possíveis são: outside,inside ou inherit.



5.3. Pseudo-elementos

Pseudo-classes e pseudo-elementos são usados em CSS, para adicionar efeitos a um seletor, ou a parte de um seletor.

A sintaxe dos pseudo-elementos é:

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

Na estilização de textos, os dois pseudoelementos mais comuns são:

- first-letter
- first-line

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

```
p:first-letter
{
font-size:300%;
color:#6666FF ;
font-weight:bold;
border-width:2px;
border-style:solid;
border-color:#009966;
}
```

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

```
p:first-line
{
color: #0000FF;
font-variant: small-caps;
text-decoration:overline;
font-family:"Courier New", Courier, monospace;
}
```

O elemento âncora juntamente com o atributo href têm a função de criar links em um documento. Para estilizarmos links em uma página, é fundamental a utilização de pseudo-classes.

As pseudo-classes são elementos que complementam seletores e a maioria das pseudo-classes são dinâmicas, ou seja, o efeito produzidos por elas depende de interação com o usuário.

São quatro as pseudoclasses para links:

Pseudoclasses	Estado do link
a:link	Link no estado inicial
a:visited	Link visitado
a:hover	Estado do link quando o usuário passa o ponteiro do mouse sobre ele
a:active	Estado do link quando o usuário clica sobre ele

Efeitos sublinhados: os navegadores normalmente apresentam links não visitados na cor em tonalidade azul, os visitados na cor púrpura e também os destacam com um sublinhado.

Para retirar o sublinhado padrão dos links, teremos que usar a propriedade text-decoration definida como o valor none.

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

Após ter retirado o sublinhado do link no seu estado inicial, é comum definir um estilo devolvendo o sublinhado quando o usuário passa o ponteiro do mouse sobre ele. Para isso usaremos as pseudoclasses hover e active.

```
a:hover,a:active
{
text-decoration:underline;
}
```

Além de alterar o sublinhado, podemos alterar cor, fundo, fonte entre outras propriedades css usando estas pseudoclasses.

Efeitos rollover: para criar um efeito rollover simples podemos definir uma cor de fundo para o estado hover do link. Veja na regra css abaixo:

```
a:link,a:visited
{
text-decoration:none;
border-width:1px;
border-color:#000000;
border-style:solid;
background-color:#009900;
color:#FFFF00
}
```

```
a:hover,a:active
{
text-decoration:none;
border-width:1px;
border-color:#000000;
border-style:solid;
background-color:#FFFF00;
color:#009900
}
```

OBSERVAÇÃO IMPORTANTE: As pseudoclasses podem ser utilizadas em qualquer seletor css. Podem ser aplicados várias regras CSS aos efeitos HOVER em links, a citar:

- Alterar plano de fundo
- Alterar tamanho da fonte do texto
- Alterar fonte

5.4. Exercícios práticos

Questão 01. Criar uma regra CSS para formatação de parágrafos utilizando as seguintes configurações:

- Borda Esquerda: Espessura de 20em, na cor verde e estilo solid.
- Borda Direita: Espessura de 20em, na cor vermelha e estilo solid.
- Largura: 60%
- Altura: 400px;
- Tamanho da fonte: 12px;
- Espaçamento entre letras: 3em;
- Espaçamento entre palavras: 4ex;
- Altura dos blocos: 3em;
- Alinhamento do texto: justificado
- Tipo de fonte Verdana
- Endentação do texto: 25px;

Por fim adicione aos pseudo-elementos **`:firstletter`** e **`:firstline`**, as seguintes configurações:

`:firstletter`

- Tamanho da Fonte: 36px
- Cor da fonte: azul
- Decoração do texto: overline

`:firstline`

- Decoração do texto: underline
- Capitalização do texto: todos os caracteres em caixa alta

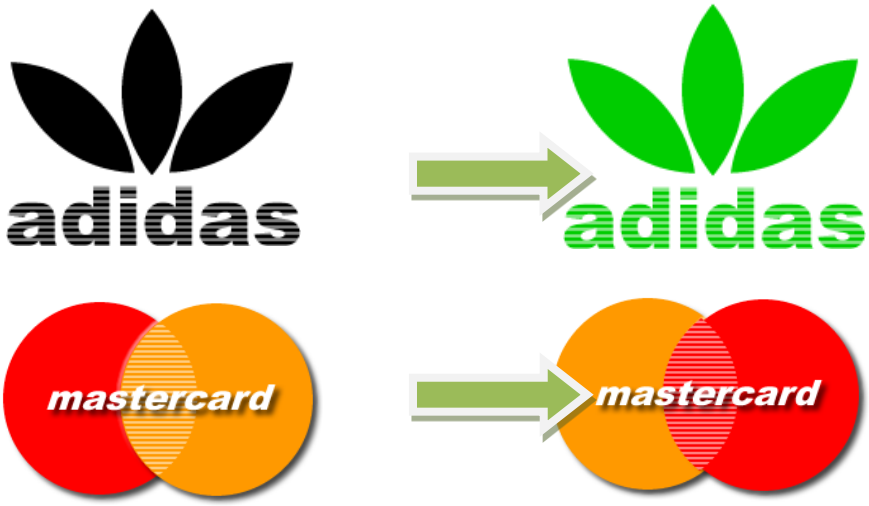
Questão 02. Criar regras css para os seguintes links:

a) Link em estado inicial sem sublinhado
b) Link em estado inicial com sublinhado acima do texto, em estado ativo com uma linha passando ao meio do texto, e em estado visitado com sublinhado abaixo do texto
c) Link em estado inicial na cor laranja e em estado hover na cor verde e sublinhado
d) Link em estado inicial com bordas com estilo solida e cor azul, fundo preto e cor da fonte branco; e em estado hover com bordas com estilo pontilhadas e cor preto, fundo branco e cor da fonte preto
e) Link em estado inicial de cor vermelha; estado visitado em itálico e negrito na cor laranja; estado hover sublinhado, e quando o usuário clicar no link a cor deverá ser amarelo.

Para cada regra acima, crie uma classe ou id e em seguida associe-as para os links criados. A regra CSS deverá ser criada de forma interna.

Questão 03.

Adicionar as seguintes imagens em uma página HTML. Adicione regras CSS para que ao passar o cursor do mouse sobre as imagens, as imagens sejam alteradas conforme a ilustração abaixo:



Questão 04. Com base neste do topo adaptado do site Globo.com, você deverá criar um código HTML para que este seja apresentado no navegador de forma semelhante a imagem a seguir.

Em seguida deverá ser criado um efeito hover para que aparecem bordas sobre cada item do menu (notícias, esportes, entretenimento) quando o usuário passar o cursor do mouse sobre cada item. Exemplo: Quando o cursor do mouse estiver sobre o item esporte, este deve ser visualizado semelhante a imagem abaixo:

Questão 05. Criar uma regra Css para listas com as seguintes características

- Utilizar pequena seta na cor vermelha e usá-la como imagem de marcador da lista

- Cor do texto vermelho
- Fonte verdana de tamanho 12px
- A borda inferior os itens do da lista serão traçadas, de espessura 1px e cor vermelha;

Após criar a regra acima, você deverá criar outra regra css (efeito hover) de forma que ao passar o mouse sobre os elementos da lista, a imagem de marcador da lista, a borda inferior o texto estejam na cor verde.

Para testar a regra acima, crie uma lista não numerada com 5 itens:

1 Kg de Batatas
2 Ovos
1 Xícara de Leite
200 gramas de farinha de trigo
1 colher de sopa de manteiga

Exemplo:

1 Kg de Batatas
2 Ovos
1 Xícara de Leite
200 gramas de farinha de trigo
1 colher de sopa de manteiga

Questão 06: O site Globo.Com utiliza um efeito hover para adicionar uma borda a uma imagem e sublinhar o texto relativo a um link que irá direcionar a uma outra página. Veja exemplo:

Utilizando o exemplo acima, crie uma página HTML que produza o mesmo efeito apresentado.

6. POSICIONAMENTO E LAYOUT COM CSS

6.1. Posicionamento

Existem quatro tipos de posicionamento em CSS: estático, relativo, absoluto e fixo. A propriedade CSS que possibilita posicionar um elemento qualquer é a propriedade **positivon**. Esta propriedade deverá receber um dos valores referentes aos quatro tipo de posicionamento CSS.

O posicionamento estático é definido pelo valor static da propriedade position. Assim: **elemento { position: static }.**

Um elemento posicionado estaticamente segue o fluxo normal dos elementos da página, ou seja, se posiciona abaixo do elemento imediatamente anterior e acima do imediatamente posterior, quando nenhum destes está posicionado de outra forma que não a estática.

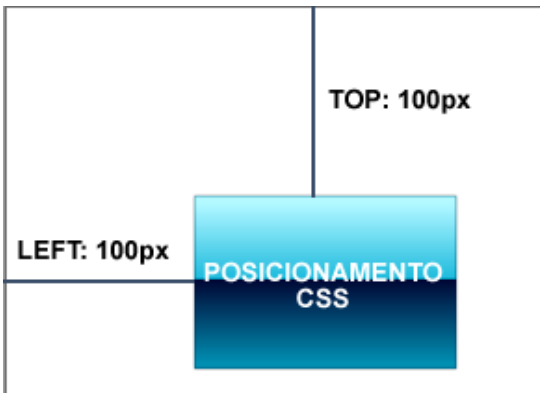
Como o posicionamento estático é o padrão, não é necessário escrever regras para esse tipo de posicionamento.

O posicionamento absoluto é definido pelo valor absolute da propriedade position. Assim: **elemento{ position:absolute }**

Quando um elemento é posicionado absolutamente, sua posição é computada de acordo com a posição do elemento "posicionado" mais próximo, que o contém. Elemento "posicionado" é qualquer elemento que tenha seu posicionamento definido como relativo, absoluto ou fixo. Quando não há nenhum elemento "posicionado", a posição é computada com relação ao elemento body.

Na regra CSS, ao definir a posição absolute utilizamos as propriedades (left) que define o quanto a imagem dista da margem esquerda e a propriedade (top) que define o quanto a imagem dista do topo.

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



Podemos também alterar as referências do elemento posicionado, mudando as propriedades para right(direita) e bottom (inferior). Dessa forma, podemos fazer combinações para os quatro cantos da tela.

Qualquer unidade de medida CSS pode ser usada para posicionar, inclusive medidas relativas como são aquelas expressas em porcentagens.

Considere a marcação HTML a seguir. Para marcação HTML foi desenvolvida duas regras CSS.

Marcação HTML

```
<img id="pos1‟ src="posicionamento.png"/>
<img id="pos2" src="posicionamento.png"/>
```

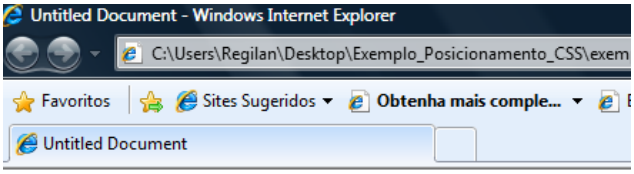
Regra CSS

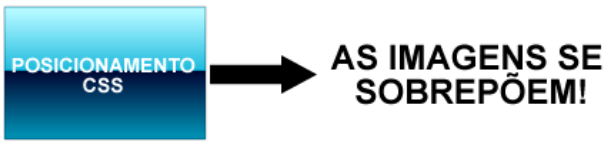
```
#pos1
{position:absolute;
top:100px;
left:0px;
}


#pos2{
position:absolute;
top:100px;
left:0px;
}
```

O que será exibido com as regras acima?

Como as duas id criadas possuem posicionamento absoluto, e o posicionamento a esquerda é de 100px, as imagens se sobrepõem, ou seja são posicionadas uma em cima da outra.

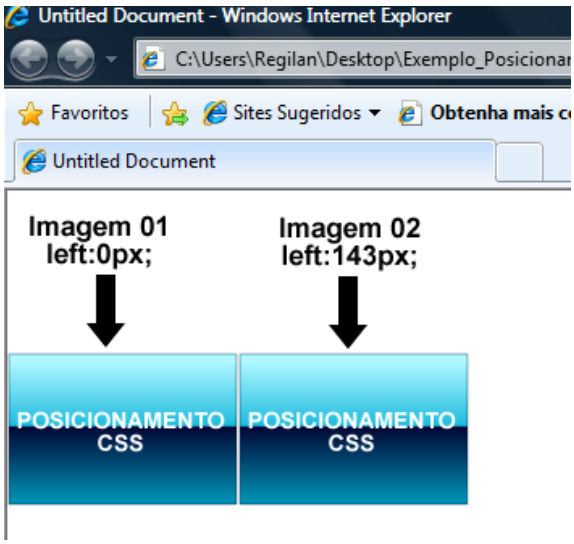




Para que as imagens não sejam sobrepostas, podemos alterar o código CSS para a seguinte forma:

```
#pos2{
position:absolute;
top:100px;
left:143px;}
```

Como a imagem apresenta largura de 143px, o posicionamento a esquerda da segunda imagem estará configurado com 143px, de forma que aparecerá logo em seguida a apresentação da primeira imagem.



Especificando **position:relative** podemos utilizar top ou bottom, right ou left para posicionar os elementos na página em relação ao lugar que ele ocuparia no fluxo do documento. Veja a regra CSS para a posição relative para o exemplo apresentado anteriormente.

```
#pos1{
position:relative;
top:100px;
left:0px;}

#pos2{
position:relative;
top:100px;
left:0px;}
```

O que será exibido com as regras acima?

Quando foi utilizado o posicionamento absoluto, as imagens apareceram sobrepostas, já que elas estavam posicionadas a 0px do elemento que as contém(o body).

Com o posicionamento relativo, a primeira imagem aparecerá a 0(zero) px do elemento anterior(body), e a segunda imagem aparecerá a 0px do elemento anterior(imagem1).

Considere a seguinte marcação HTML:

```
<div id="div1">
<div id="div2"></div>
```

```
    </div>
```

Se definirmos uma regra CSS position: relative para a div1 e position: absolute para a div2, a partir de agora a div2 se posicionará absolutamente em relação a div1, não mais em relação ao body da página.

Dessa forma ao configurarmos as propriedades: left, top, bottom e right, elas serão configuradas em relação a div1.

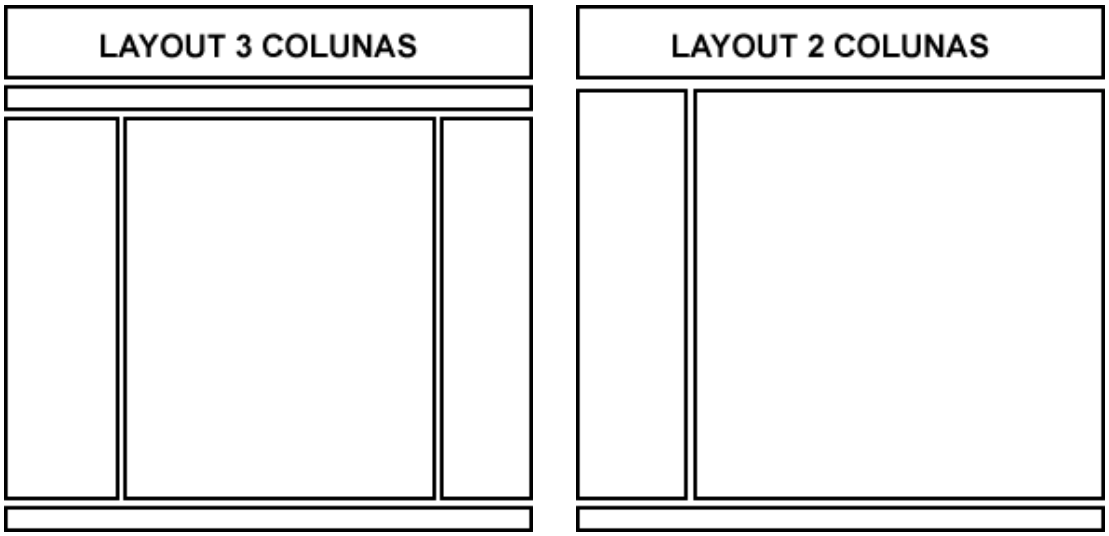
O esquema de posicionamento **FIXO**, posiciona um elemento fixadamente na página em relação a janela, ou seja, independente de a pagina conter um scroll ou não, o elemento sempre ficará visível ao usuário em relação as medidas definidas.

O posicionamento fixo é definido pelo valor fixed da propriedade position. Assim: elemento {position:fixed }.

Da mesma forma que o posicionamento relativo e absoluto, deverá também ser configurado as propriedades left,right,top e bottom.

6.2. Layout

O primeiro passo para construir um layout css é definir como será o design da página. O design de uma página pode ser feito em ferramentas(Fireworks, CorelDrawn, Photoshop) apropriadas pra criação de design de páginas WEB ou mesmo em uma folha de papel.



Em nosso exemplo construiremos um layout 3 colunas. A estrutura HTML para definir as seções do layout consistirá de divs e para cada div será definida regras CSS para posiciona-las.

```
<div id="containerGeral">
    <div id="cabecalho"> </div>
    <div id="menu"> </div>
    <div id="le"> </div>
    <div id="centro"> </div>
    <div id="ld"> </div>
    <div id="rodape"> </div>
</div>
```

O código do containerGeral e do cabeçalho ficará assim:

```
#containerGeral
{
     position:relative;
     width:800px;
     height:auto;
     border:1px solid #000;
     margin:0 auto;
}
```

```
#cabecalho
{
     width:800px;
     height:130px;
     background-color:#06F;
}
```

O containerGeral deverá ter uma regra de posicionamento. Usaremos o posicionamento do fluxo do documento, posicionamento relative.

A regra margin: 0 auto; configurará a margem superior e inferior para o valor 0 e a esquerda e direita para automática, neste caso a página ficará centralizada.

O menu, rodapé e lateral esquerda ficará assim:

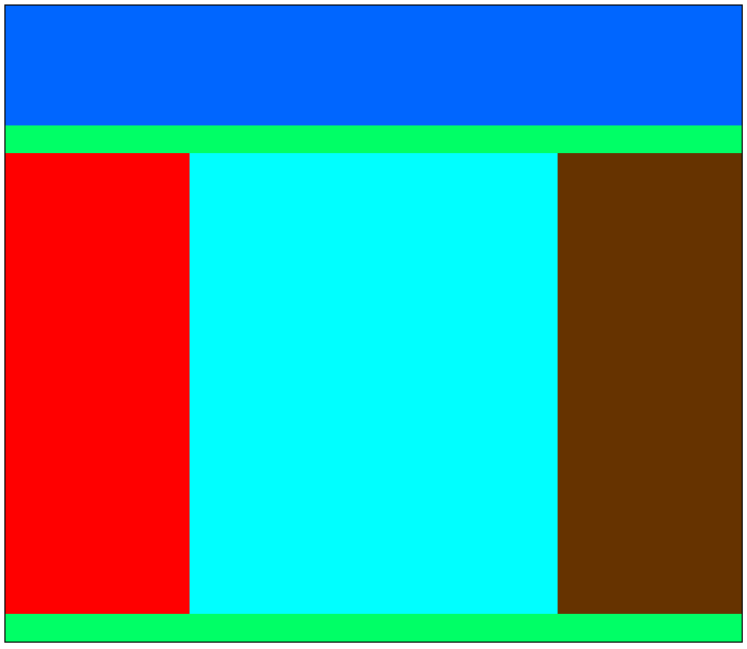
```
#menu,#rodape
{
     width:800px;
     height:30px;
     background-color:#0F6;
}
```

```
#le
{
	width:200px;
	height:500px;
	background-color:#F00;
}
```

O centro e a lateral direita possui a seguinte codificação CSS

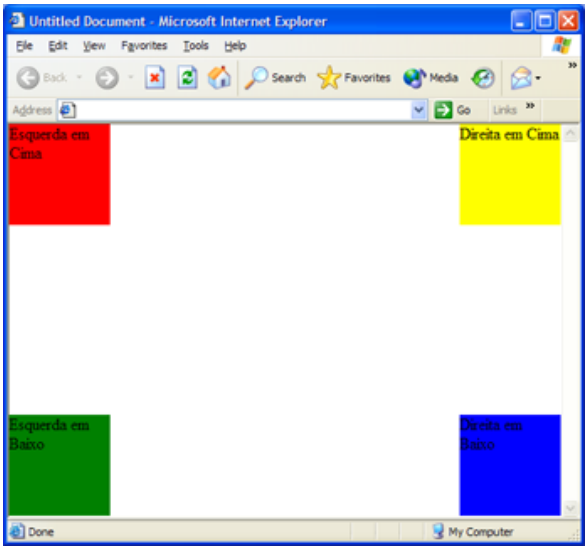
```
#centro
{
	width:400px;
	height:500px;
	background-color:#0FF;
	position:absolute;
	top:160px;
	left:200px;
}
#ld
{
	width:200px;
	height:500px;
	background-color:#630;
	position:absolute;
	top:160px;
	left:600px;
}
```

Resultado:



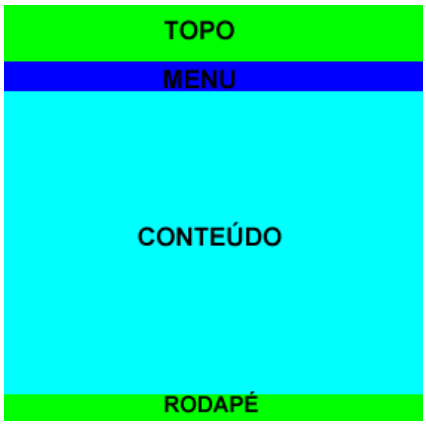
6.3. Exercícios práticos

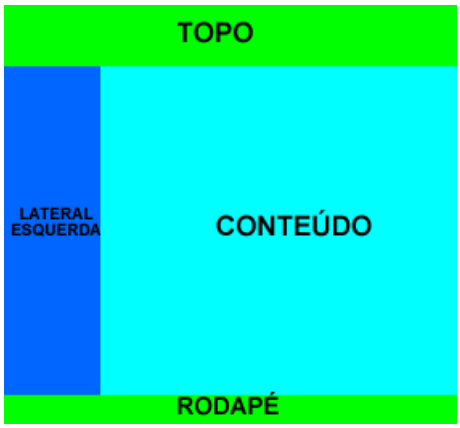
Questão 01. Desenvolva um arquivo HTML com 4 caixas conforme apresentado a seguir; e utilizando regras CSS de posicionamento permita produzir o seguinte resultado:

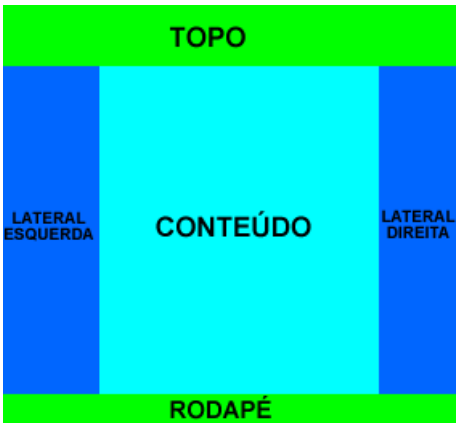


Questão 02. Utilizando Divs e regras CSS, desenvolva os seguintes layouts:

a) **b)** **c)**







Questão 03. Utilizando os recursos HTML e CSS já estudados e regras de posicionamento, crie páginas HTML para apresentar o resultado mostrados nas imagens a seguir.

a)

DICAS

Aprenda a fazer catchup em casa

DOCE

Confira receita de minipudim de chocolate

APRENDA A FAZER

Lasanha bem leve com berinjela e ricota

b)

Compre Agora!

Adidas Komet

Economize R$ 150

Só 10x R$ 19,99

Oferta Dell

i14 Série 3000

Intel® Core™ i3

10x de R$ 149,90

Ofertas Netshoes

Camisa Polo Vasco Expresso ...

4x de R$ 27,48

c)

Eleições 2014

VEJA A COBERTURA COMPLETA DA CAMPANHA

DATAFOLHA

Dilma sobe, Marina fica estável e Aécio cai

POLÍTICA

Lula indica a possibilidade de ser o candidato do PT nas eleições de 2018

- Pezão: Discurso de Garotinho anima traficantes
- Por que Marina tergiversa tanto? Porque está na Bíblia

PODER ONLINE

PT vê em Lula chave para evitar dissidências na direção

MAIS DATAFOLHA

Alckmin mantém favoritismo em SP, mas vantagem sobre Skaf cai

- Pimentel tem 32% e Pimenta aparece com 24%
- Garotinho tem 28%, Pezão, 23% e Crivella, 18% no Rio

CAMPANHA

Campanha contra cavaletes de candidatos ganha as redes sociais

BLOG DO KENNEDY

Recuos freiam subida de Marina; PT vê situação desesperadora em SP

7. FORMULÁRIOS WEB

Um dos recursos mais interessantes da linguagem Html é a possibilidade de criar formulários eletrônicos.

Usando um formulário o usuário pode interagir com o servidor, enviando dados que serão processados no servidor e posteriormente devolvidos ao cliente.

Esses comandos são os principais responsáveis pela viabilização da troca de informações entre o cliente e o servidor. Eles podem ser usados em qualquer tipo de atividade.

Os formulários são iniciados com a tag de abertura <form> e encerrados com a tag de fechamento </form>.

É muito importante que todo o conteúdo do corpo do formulário esteja entre essas duas tags de abertura e fechamento.

```
<form> </form>
```

Os formulários podem conter qualquer formatação - parágrafos, listas, tabelas, imagens - exceto outros formulários.

Em especial, colocamos dentro da marcação de <FORM> as formatações para campos de entrada de dados, que são três:

- <input>
- <select>
- <textarea>

A forma mais simples de campo de entrada é a marcação text utilizando a tag <input>.

A tag <input> tem um atributo TYPE, ao qual atribuímos seis valores diferentes para gerar seis tipos diferentes de entrada de dados:

- Text
- Radio
- Checkbox
- Password
- Hidden
- File

Além do atributo type, a tag <input> apresenta os seguintes atributos:

- O atributo name identifica o campo através de um nome
- O atributo size representa a largura da caixa
- O maxlength representa o número máximo de caracteres a serem digitados
- O atributo value representa o conteúdo da caixa de texto

Quando INPUT não apresenta atributos, é assumido que TYPE=TEXT (valor padrão);

Veja a marcação HTML:

```
Nome: <input type=text name="nome“>
```

ou

```
Nome: <input name="nome“>
```

A marcação acima produz o resultado:

Configurando o atributo type com o valor password, a entrada de texto passa a ter os caracteres escondidos por asteriscos, como se pode ver no exemplo:

```
Login: <input type=text name="login"><br>
Password: <input type=password name="senha">
```

A marcação acima produz o resultado:

Esse é um campo que permite ao usuário a escolha de várias opções diante de uma série de alternativas.

```
<input type=“checkbox”>
```

O <input type=“checkbox”> aceita múltiplas seleções, logo a pergunta feita ao usuário pode ter mais de uma resposta possível.

Considere a marcação HTML abaixo:

```
     Esportes que você pratica:<br>

     <input name="esporte" value="futebol"
type="checkbox">Futebol<br>
     <input name="esporte" value="volei"
type="checkbox">Vôlei<br>
     <input name="esporte" value="natacao"
type="checkbox">Natação<br>
     <input name="esporte" value="basquete"
type="checkbox">Basquete<br>
     <input name="esporte" value="tenis"
type="checkbox">Tênis<br>
     <input name="esporte" value="bocha"
type="checkbox">Atletismo<br>
```

Note que o nome do campo (NAME) é o mesmo para toda a lista de valores.

Além disso pode ser usado o atributo CHECKED, que marca uma escolha inicial, isto é, se o usuário não escolher nenhuma opção, o formulário irá considerar a alternativa "pré-escolhida".

A marcação HTML apresentanda anteriormente produz o seguinte resultado

Esportes que você pratica:
- ☐ Futebol
- ☐ Vôlei
- ☐ Natação
- ☐ Basquete
- ☐ Tênis
- ☐ Atletismo

Caso você queira adicionar o atributo checked para o esporte natação, a marcação HTML é:

```
    <input name="esporte" value="natacao" type="checkbox"
checked="checked">Natação<br>
```

O <input type=“radio”> não aceita múltiplas seleções como o checkbox, logo a pergunta feita ao usuário e que será respondida através da seleção de um dos campos rádio só poderá ter uma resposta possível.

Considere a marcação HTML abaixo:

```
    Você tem mais de 30 anos?<br/>
    <form>
    Sim <input type="radio" name="cor" value="sim">
    Não <input type="radio" name="cor" value="nao">
    </form>
```

A marcação HTML do slide anterior, produz o seguinte resultado:

Você tem mais de 30 anos?
Sim ○ Não ○

Assim como acontece na checkbox, pode ser usado o atributo CHECKED, que marca uma escolha inicial, isto é, se o usuário não escolher nenhuma opção, o formulário irá considerar a alternativa "pré-escolhida".

A marcação TEXTAREA não utiliza o formato convencional INPUT TYPE="text" dos exemplos anteriores.

Ao contrário, uma marcação <TEXTAREA> delimita o seu início e a marcação </TEXTAREA> o seu fim.

```
    <textarea> </textarea>
```

O tamanho da caixa de texto é determinada pelos atributos de colunas (cols) e de linhas (rows).

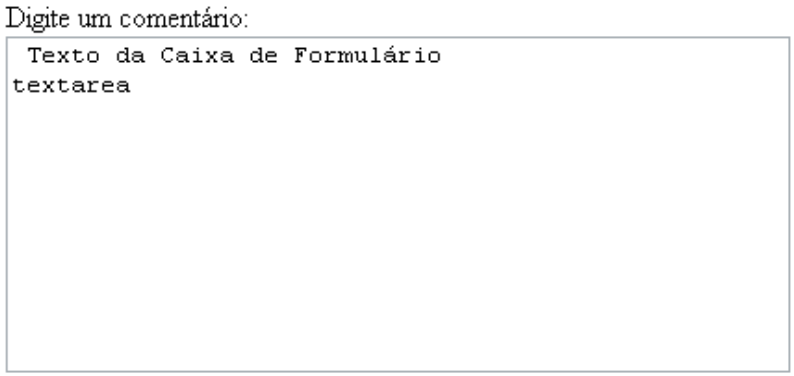
O atributo name é responsável por identificar esse campo.

Comentários:

Considere a marcação HTML abaixo:

```
     Digite um comentário: <br>
     <form>
     <textarea name="identificador" rows="10" cols="50"> Texto
da Caixa de Formulário
     textarea </textarea>
     </form>
```

A marcação acima, produz o seguinte resultado:



As tags <select>...</select> apresentam uma lista com diversos itens que podem ser selecionados.

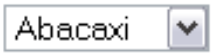
É possível configurar essa tag para aceitar uma seleção simples ou múltiplas seleções.

Dentro de <select>...</select> fica as tags <option> responsáveis por apresentarem os itens da lista, e o atributo value guarda o valor quando o item é selecionado.

Veja a marcação HTML abaixo:

```
<select name="sabor">
<option>abacaxi</option>
<option>creme</option>
<option>morango</option>
<option>chocolate</option>
</select>
```

A marcação acima, produz o seguinte resultado:



A tag <option> representa os itens da lista, que será visível assim que o usuário clicar na lista

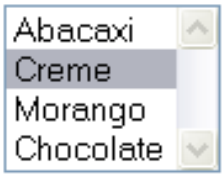
Também é possível estabelecer uma escolha-padrão, através do atributo SELECTED

```
<option selected> creme </option>
```

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

```
<select name="sabor" size=4>
```

Veja o resultado da marcação acima:



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

```
<select name="sabor" size=4 multiple>
```

A tag responsável pela criação do botão de envio é o <input> e os seus atributos são:

- type:
  - type=”submit” – indica que a função desse botão e de enviar o formulário.
  - type=”reset” – indica que os dados preenchidos serão todos apagados e o formulário não será enviado.
  - type=”button” – esse type não tem nenhuma função e normalmente é utilizado em conjunto com um comportamento de Javascript que será acionado pelo evento onClick.
- text: texto que aparecerá no botão.
- name= texto para identificar esse input.

Considere a marcação HTML:

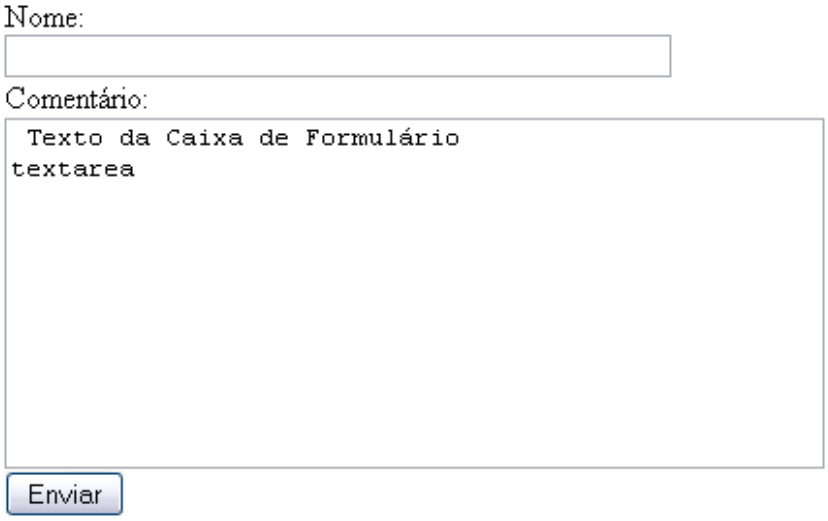
```
<input type="submit" value="Enviar">
```

O código acima produz o seguinte resultado

Considere a marcação HTML abaixo:

```
<form>
Nome:<br> <input type="text" size="53"> <br>
Comentário:<br> <textarea name="identificador" rows="10"
cols="50"> Texto da Caixa de Formulário
textarea </textarea>
<br>
<input type="submit" value="Enviar">
</form>
```

A marcação HTML apresentada no slide anterior produz o seguinte resultado:

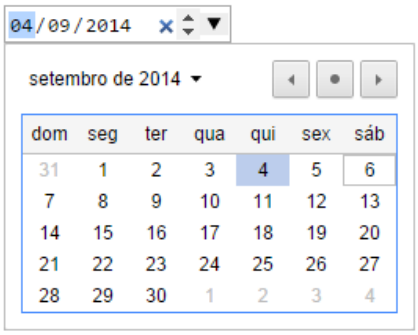


Caso seja necessário substituir o botão de submit por uma imagem que tenha a mesma função, é possível utilizando a tag: <input type=”image”>

```
<input type=”image” src=”local da imagem” alt=”texto” border=”0”
>
```

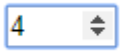
Alguns novos tipos de componentes de entrada surgiram com a HTML 5, listaremos a seguir alguns dos principais:

date:




```
<input type=”date”/>
```

number:




```
<input type="number" min="1" max="5">
```

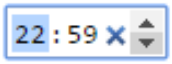
range:

```
<input type="range" min="0" max="10">
```

search:


```
<input type="search" />
```

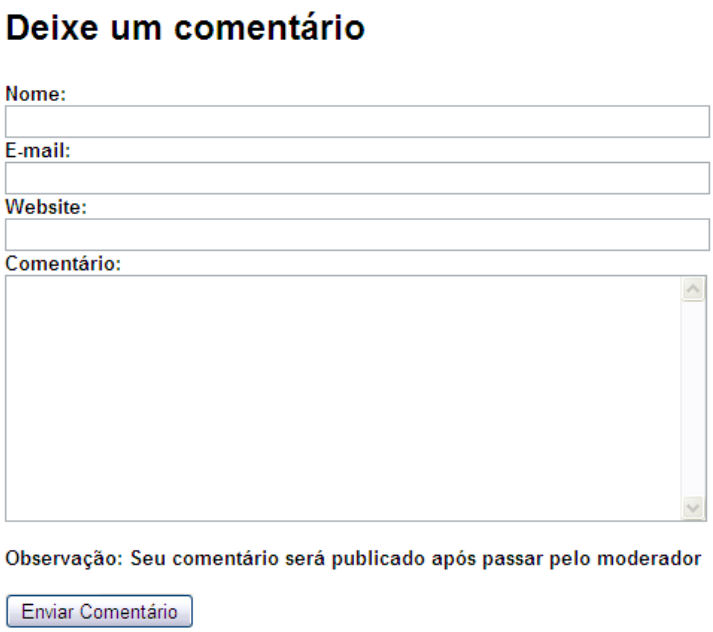
time:



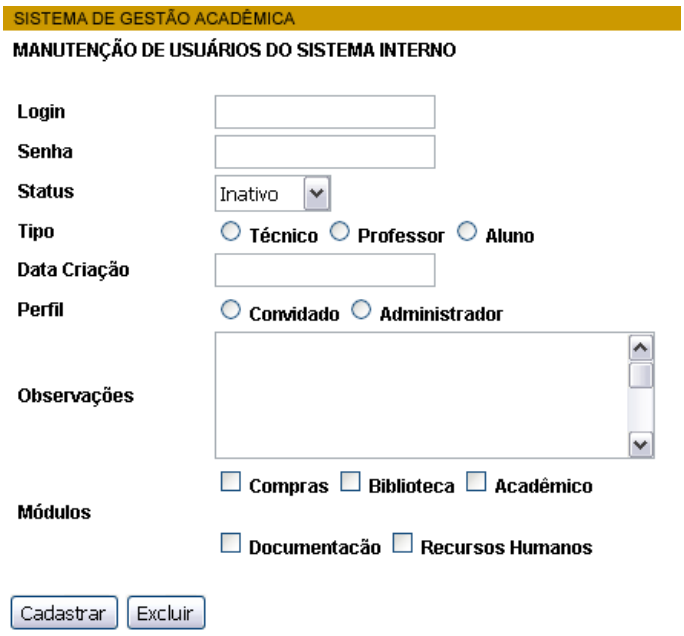

```
<input type=“time" />
```

7.1. Exercícios práticos

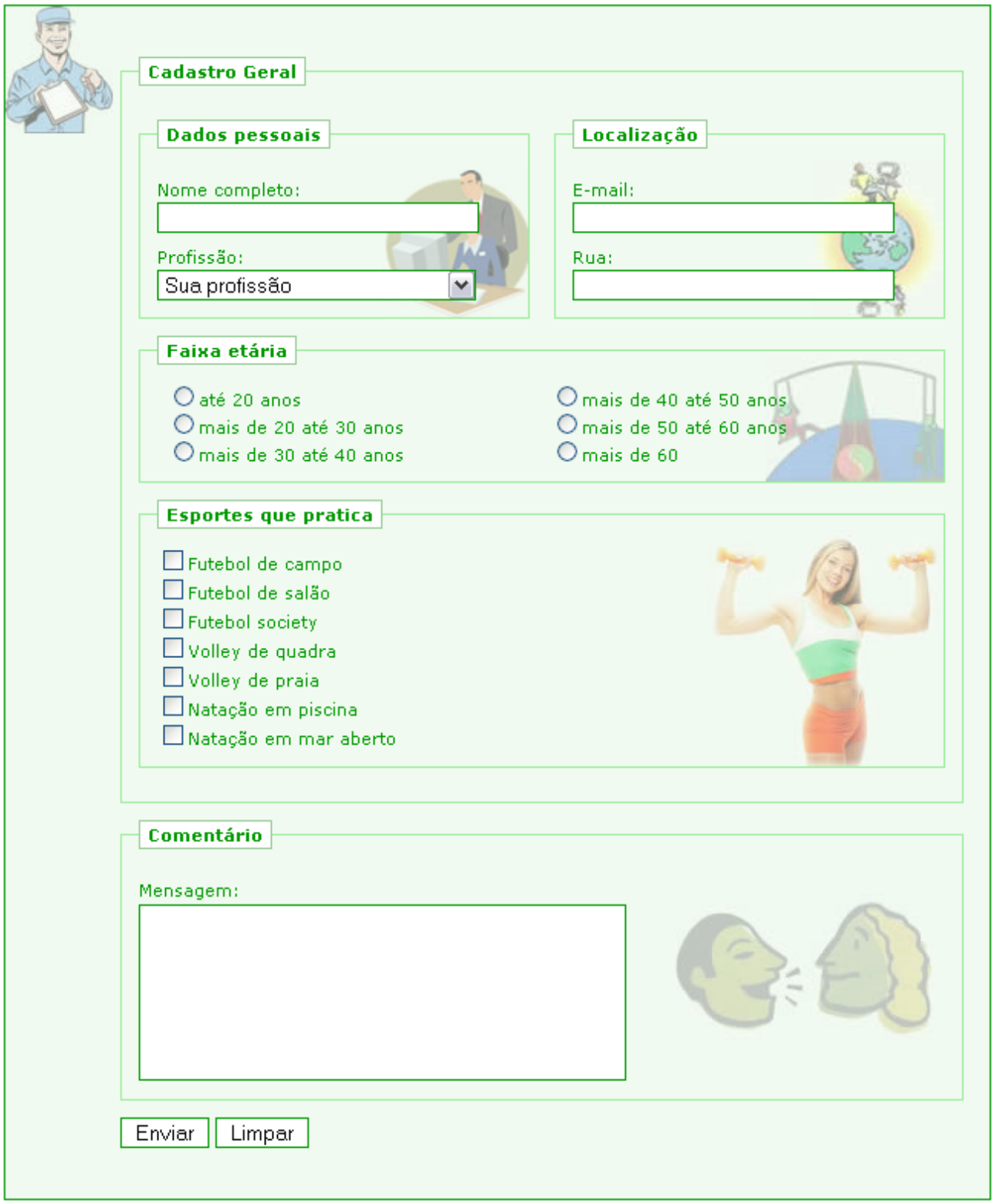
Questão 01. O formulário a seguir é um formulário padrão para comentários em Blogs. Utilize as tags de formulário para desenvolvê-lo conforme abaixo:



Questão 02. O formulário abaixo representa uma página de cadastro de um usuário em um sistema interno de uma universidade. Utilize as tags de formulário para desenvolvê-lo conforme abaixo:

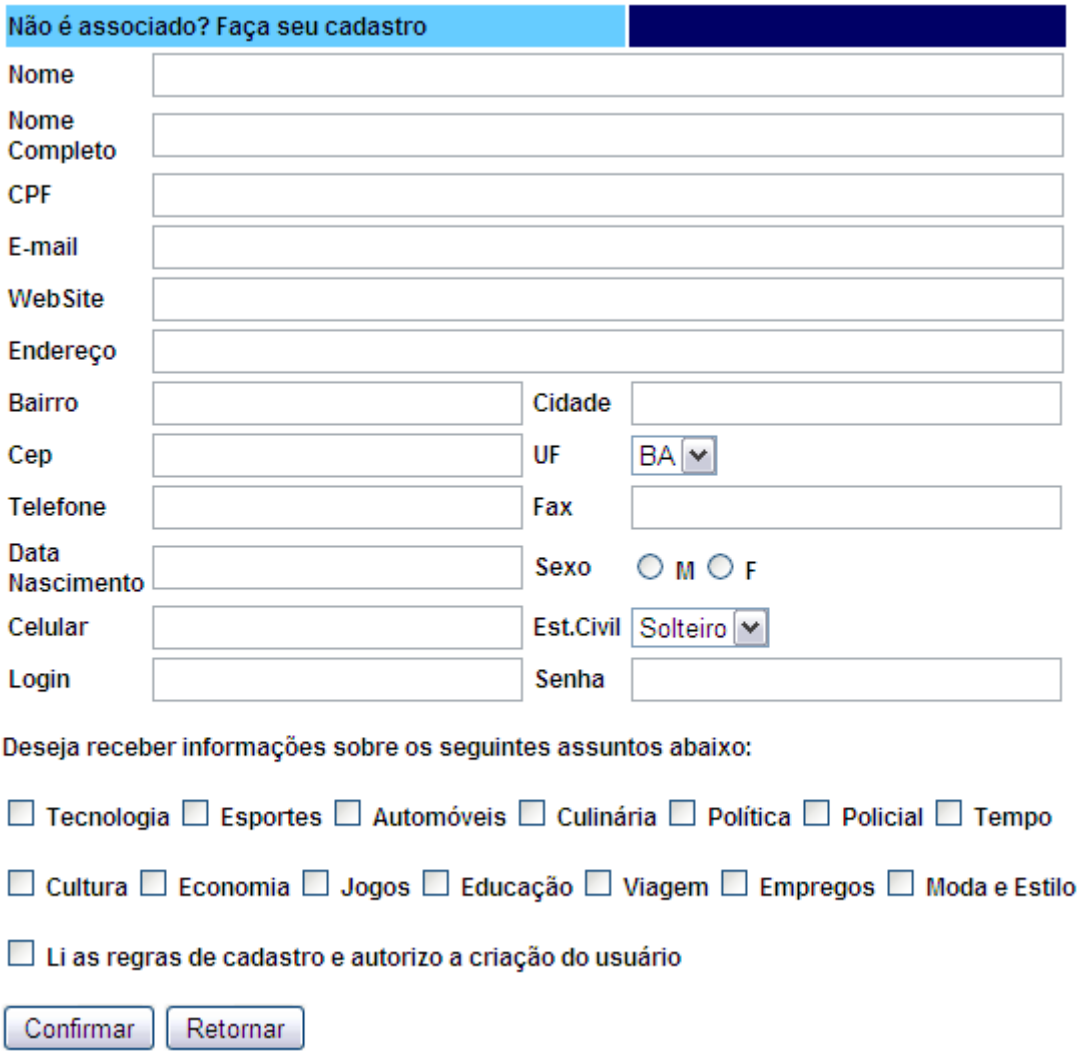


Questão 03. Construir um formulário em HTML e aplicar as regra css de forma que fique semelhante a imagem abaixo:



Questão 04. Desenvolver o seguinte formulário de cadastro com as seguintes características.

- A célula com o texto: “Não é associado? Faça seu cadastro” deverá ter cor de fundo azul claro com texto na cor preta. A outra célula da mesma linha deverá ter cor de fundo azul escuro
- O campo UF, deverá vir com as siglas de todos os Estados Brasileiros para o usuário escolher
- O campo Est. Civil deverá vir com os seguintes dados: Solteiro, Casado, Viúvo e Divorciado
- O campo senha deverá ter os dados escondidos. Ou seja, ao digitar os valores para preencher o campo ele deverá está no formato de “password”



OBS: Todas as imagens utilizadas na página acima estão em uma pasta compactada junto com a atividade.

8. JAVASCRIPT

8.1. Noções Gerais

JAVASCRIPT É UMA LINGUAGEM de programação interpretada criada em 1995, como uma extensão do HTML para o browser Navigator 2.0. Hoje existem implementações JavaScript nos browsers dos principais fabricantes.

Programas em JavaScript são interpretados linha-por-linha enquanto o browser carrega a página ou executa uma rotina.

JavaScript é baseada em objetos. Trata suas estruturas básicas, propriedades do browser e os elementos de uma página HTML como objetos (entidades com propriedades e comportamentos) e permite que sejam manipulados através de eventos do usuário programáveis, operadores e expressões.

Com JavaScript pode-se fazer diversas coisas que não é possível com HTML:

- Realizar operações matemáticas e computação.
- Abrir janelas do browser, trocar informações entre janelas, manipular propriedades do browser como o histórico, barra de estado, plug-ins e applets.
- Interagir com o conteúdo do documento, alterando propriedades da página, dos elementos HTML e tratando toda a página como uma estrutura de objetos.
- Interagir com o usuário através do tratamento de eventos

Para editar código JavaScript, é necesspario apenas um simples editor de texto, como o Bloco de Notas do Windows.

Pode-se também usar um editor HTML. Alguns editores colocam cores ou dão destaque ao código JavaScript. Outros até permitem a geração de código ou a verificação de sintaxe.

Há três maneiras de incluir JavaScript em uma página Web:

- Dentro de blocos HTML <script> ... </ script > em várias partes da página.
- Em um arquivo externo, importado pela página: para definir funções que serão usadas por várias páginas de um site.
- Dentro de descritores HTML sensíveis a eventos: para tratar eventos do usuário em links, botões e componentes de entrada de dados, durante a exibição da página.

A forma mais prática de usar JavaScript é embutindo o código na página dentro de um bloco delimitado pelos descritores HTML <script> e </script>.

Pode haver vários blocos <script> em qualquer lugar da página.

```
<script>
... instruções JavaScript ...
</script>
```

O descritor <script> possui um atributo language que informa o tipo e versão da linguagem utilizada. O atributo language é necessário para incluir blocos em outras linguagens como VBScript. É opcional para JavaScript:

```
<script language="vbscript"> ...código em vbscript... </script>
<script language="javascript"> ...código javascript... </script>
<script> ... código
```

Muitas vezes é necessário realizar um mesmo tipo de tarefa mais de uma vez. Para esse tipo de problema JavaScript permite que o programador crie funções que podem ser chamadas de outras partes da página várias vezes.

Se várias páginas usam as mesmas funções JavaScript definidas pelo autor da página, uma boa opção é colocá-las em um arquivo externo e importá-lo nas páginas que precisarem delas.

Este arquivo deve ter a extensão ".js" e conter apenas código JavaScript (não deve ter descritores HTML, como <script>). Por exemplo, suponha que o arquivo biblio.js possua o seguinte código JavaScript:

```
function soma(a, b)
{
     return a + b;
}
```

Para carregar esta função e permitir que seja usada em outra página, usa-se o atributo src do descritor <script>:

```
<script LANGUAGE=JavaScript SRC="biblio.js" >
resultado = soma(5, 6);
document.write('<p>A soma de 5 e 6 é ' + resultado</p>');
</script>
```

A linguagem JavaScript permitem a captura de eventos disparados pelo usuário, como o arrasto de um mouse, o clique de um botão, etc. Quando ocorre um evento, um procedimento de manuseio do evento é chamado. O que cada procedimento irá fazer pode ser determinado pelo programador.

Os atributos de eventos se aplicam a elementos HTML específicos e e sempre começam com o prefixo “on”. Os valores recebidos por esses atributos é código JavaScript. Por exemplo:

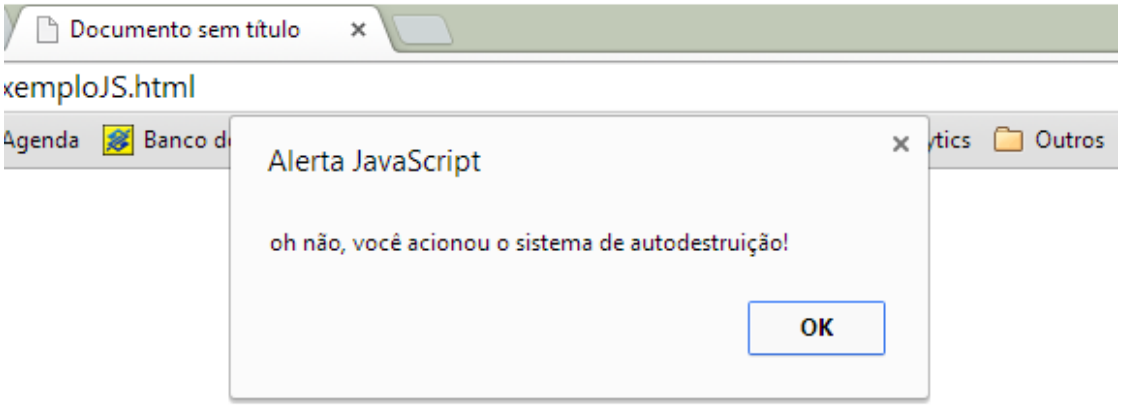
```
<form>
<input type="button"
onclick="alert('oh não, você acionou o sistema de
autodestruição!')"
value="não aperte este botão">
</form>
```

Como relação ao código acima, tudo o que aparece entre as aspas duplas do atributo onclick é JavaScript. onclick é um atributo HTML, criado como extensão para dar suporte ao evento de clicar o botão.

O código JavaScript que está em negrito será interpretado quando o usuário apertar o botão com o mouse (onclick).

A instrução alert() cria uma janela de alerta (acima) com a mensagem passada como parâmetro (entre parênteses e aspas no código em negrito).

O código acima, quando executado no navegador aparecerá semelhante a imagem abaixo:



Os procedimentos de manuseio de eventos introduzidos por JavaScript são:

Atributo HTML	Quando o procedimento é executado	Descritores HTML onde é suportado
onclick	Quando um objeto é clicado pelo mouse	<a>, <input>
onselect	Quando um objeto é selecionado	<input type=text>, <textarea>
onchange	Quando uma seleção ou campo de texto tem seu conteúdo modificado	<input type=text>, <textarea>, <select>
onfocus	Quando um componente de formulário ou janela se torna ativa	<textarea>, <body>, <form>, <input>, <select>, <option>
onblur	Quando um componente de formulário ou janela se torna inativa	<textarea>, <body>, <form>, <input>, <select>, <option>
onmouseover	Quando o mouse está sobre um link	<a>, <area>
onmouseout	Quando o mouse deixa um link	<a>, <area>
onsubmit	Antes de enviar um formulário	<input type=submit>
onreset	Antes de limpar um formulário	<form>
onload	Após carregar uma página, janela ou frame	<body>
onunload	Ao deixar uma página, janela ou frame	<body>
onerror	Quando um erro ocorre durante a carga de uma imagem ou página	<img>, <body>
onabort	Quando a carga de uma imagem é abortada	<img>

Como procedimentos de eventos são atributos do HTML (e não do JavaScript), tanto faz escrevê-los em letras maiúsculas ou minúsculas. Usar onclick, ONCLICK ou OnClick não faz diferença. Já o texto dentro das aspas é JavaScript, que é uma linguagem que diferencia letras maiúsculas de minúsculas, portanto alert não é a mesma coisa que ALERT.

A seguir, duas funções básicas de JavaScript:

Funcão	Objetivo
document.write('mensagem')	Escreve uma mensagem na página web aberta.
alert('mensagem')	Exibe uma janela de mensagem de alerta na página

8.2. Sintaxe e estrutura

Como a maior parte das linguagens de programação, o código JavaScript é expresso em formato texto. O texto do código pode representar instruções, grupos de instruções, organizadas em blocos e comentários. Dentro das instruções, pode-se manipular valores de diversos tipos, armazená-los em variáveis e realizar diversas de operações com eles.

Instruções compostas (seqüências de instruções que devem ser tratadas como um grupo) são agrupadas em blocos delimitados por chaves ({ e }), como mostrado abaixo:

```
function xis() {
var x = 0;
while (x < 10) {
x = x + 1;
}
}
```

Blocos são usados em JavaScript na definição de funções e estruturas de controle de fluxo. Blocos são tratados como uma única instrução e podem ser definidos dentro de outros blocos.

No exemplo anterior, o bloco da função xis() contém duas instruções. A segunda é um bloco (while) que contém uma instrução. As outras instruções não pertencem a bloco algum.

As chaves que definem um bloco são caracteres separadores. Podem ser colocadas em qualquer lugar após a declaração da estrutura que representam. Não precisam estar na mesma linha. Pode haver qualquer número de caracteres terminadores de linha antes ou depois:

```
function media(a, b)
{
return (a + b)/2;
}
```

Os formatos maiúsculo e minúsculo de um caractere são considerados caracteres distintos em JavaScript.

Por exemplo function, Function e FUNCTION são três nomes distintos e tratados diferentemente em JavaScript.

Há duas formas de incluir comentários em JavaScript:

- Qualquer texto que aparece depois de duas barras (//) é ignorado pelo interpretador até o final da linha.

- Quando o interpretador encontra os caracteres /*, ignora tudo o que aparecer pela frente, inclusivecaracteres de nova-linha, até encontrar a seqüência */.

Comentários:

```
/* Esta função retorna a
* média dos argumentos passados
*/
function media(a, b)
{
return (a + b)/2; // a e b devem ser números
}
```

Variáveis são usadas para armazenar valores temporários na maior parte das instruções em JavaScript. Para definir uma variável, basta escolher um nome que não seja uma palavra reservada e lhe atribuir um valor:

```
preco = 12.6;
produto = "Livro";
```

Uma variável também pode ser declarada sem que receba um valor. Para isto é necessário usar a palavra-chave var:

```
var preco;
```

A linguagem não é rigorosa em relação a tipos de dados e portanto não é preciso declarar os tipos das variáveis antes de usá-las, como ocorre em outras linguagens.

O tipo de dados é alocado no momento da inicialização, ou seja, se na definição de uma variável ela receber uma string, JavaScript a tratará como string até que ela receba um novo tipo através de outra atribuição

O escopo ou alcance de uma variável depende do contexto onde é definida ou declarada. Uma variável declarada ou definida pela primeira vez dentro de um bloco tem escopo local ao bloco e não existe fora dele. Variáveis declaradas ou definidas fora de qualquer bloco são globais e são visíveis em todo o programa ou página HTML.

Exemplo variáveis globais:

```
<script>
global = 3; // escopo: toda a pagina
function func() {
local = 5; // escopo: somente o bloco atual
global = 10;
```

```
}
// local nao existe aqui.
// global tem valor 10! (pode ser lida em qualquer lugar da
pagina)
</script>
```

O uso de var é opcional na definição de variáveis globais. Variáveis locais devem ser definidas com var para garantir que são locais mesmo havendo uma variável global com o mesmo nome, por exemplo:

```
g = 3; // variável global
function func() {
var g = 10; // esta variável g é local!
}
// g (global) tem o valor 3!
```

Identificadores JavaScript são os nomes que o programador pode escolher para variáveis e funções definidas por ele. Esses nomes podem ter qualquer tamanho e só podem conter caracteres que sejam:

- números (0-9)
- letras (A-Z e a-z)
- caractere de sublinhado (_)

Além disso, embora identificadores JavaScript possam conter números, não podem começar com número. Existem ainda algumas palavras que são reservadas da linguagem, e estas não podem ser usadas como identificadores de variáveis

JavaScript possui várias classes de operadores. Operações de atribuição, aritméticas, booleanas, comparativas e binárias em JavaScript são realizadas da mesma forma que em outras linguagens estruturadas como C++ ou em Java.

Operador	Função
++n,n++	Incremento
--n,n--	Decremento
*	Multiplicação
/	Divisão
%	Resto
+	Adição e concatenação
-	Subtração

Operador	Função
!=	Diferente de
==	Igual a
>	Maior que
<	Menor que
>=	Maior ou igual a
<=	Menor ou igual a
||	Ou (or)
&&	E (and)
!	Negação (not)

As estruturas de controle de fluxo são praticamente as mesmas utilizadas em outras linguagens estruturadas populares.

if..else: A estrutura if... else é utilizada para realizar controle de fluxo baseado em expressões condicionais:

```
if (condição) {
// instruções caso condição == true
} else if (condição 2) {
// instruções caso condição 2 == true
} else {
// instruções caso ambas as condições sejam false
}
```

Exemplo:

```
if (ano < 0) {
alert("Digite um ano D.C.");
} else if ( ((ano % 4 == 0) && (ano % 100 != 0)) || (ano % 400
== 0)) {
alert(ano + " é bissexto!");
} else {
alert(ano + " não é bissexto!");
}
```

for: As estruturas for e while são usadas para repetições baseadas em condições. O bloco for contém de três parâmetros : uma inicialização, uma condição e um incremento. A sintaxe é a seguinte:

```
for(inicialização; condição; incremento) {
// instruções a serem realizadas enquanto condição for true
}
```

Por exemplo:

```
for (i = 0; i < 10; i = i + 1) {
document.write("<p>Linha " + i + "</p>);
}
```

A primeira coisa realizada no bloco for é a inicialização e é feita uma vez apenas. A condição é testada cada vez que o loop é reiniciado. O incremento é realizado no final de cada loop.

while: O mesmo que foi realizado com for pode ser realizado com uma estrutura while, da forma:

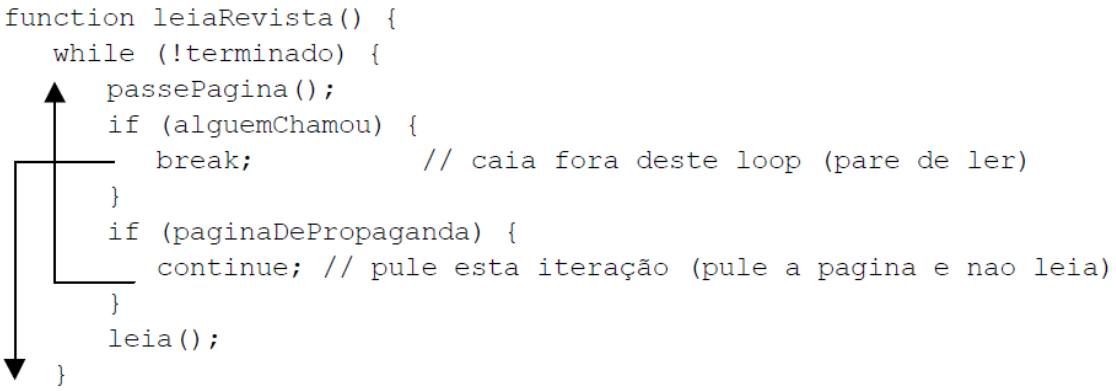
```
while(condição) {
// instruções a serem realizadas enquanto condição for true
incremento;
}
```

Veja como fica o o exemplo de repetição mostrado através do comando for e agora usando while:

```
i = 0
while (i < 10) {
document.write("<p>Linha " + i + "</p>");
i++;
}
```

- **break e continue:** Para sair a força de loops em cascata existem ainda as instruções break e continue.
- **break :** sai da estrutura de loops e prossegue com a instrução seguinte.
- **continue:** deixa a execução atual do loop e reinicia com a passagem seguinte.

Exemplo break e continue:



Em JavaScript também temos a estrutura Array(vetores).O tipo Array representa coleções de qualquer tipo, na forma de vetores ordenados e indexados. Para criar um novo vetor em JavaScript, é preciso usar o operador new e o construtor Array():

```
direcao = new Array(4);
```

Vetores começam em 0(zero) e terminam em length–1. length é a única propriedade do tipo Array. Esta propriedade ontém um número com o comprimento do vetor. Os elementos do vetor são acessíveis através de índices passados entre colchetes ([e]).

Para acessar qualquer um dos elementos do vetor, por exemplo, usa-se o nome da variável seguida do índice do elemento entre colchetes:

```
x = direcao[2]; // copia o conteúdo do terceiro elemento de
direcao em x
```

8.3. Manipulando objetos HTML com JavaScript

Todos os objetos criados em HTML estão automaticamente disponíveis em JavaScript, mesmo que um nome não seja atribuído a eles.

Por exemplo, se há três blocos <form>...</form> em uma página, há três objetos do tipo Form no JavaScript

Se eles não tem nome, pode-se ter acesso a eles através da propriedade forms definida em document.

Essa propriedade armazena os objetos Form em uma coleção ordenada de propriedades (vetor). Todos os índices usados nos vetores em JavaScript iniciam a contagem em 0, portanto, document.forms[0], refere-se ao primeiro formulário de uma página.

Cada formulário pode então ser recuperado através de seu índice:

```
frm1 = document.forms[0];
frm2 = document.forms[1];
```

Se houver, por exemplo, dentro de um bloco <form>...</form> 5 componentes, entre botões, campos de texto e caixas de seleção, existirão 5 objetos em JavaScript dos tipos Text, Button e Select.

Independente do tipo de componente de formulário, eles podem ser acessados na ordem em que aparecem no código, através da propriedade elements, de Form:

```
texto = document.forms[0].elements[1];
```

Os vetores são necessários apenas quando um objeto não tem nome. Se tiver um nome (definido no código HTML, através do atributo NAME do descritor correspondente), o ideal é usá-lo já que independe da ordem dos componentes, e pode fornecer mais informações como por exemplo, o tipo do objeto

Exemplo de manipulação de objetos do HTML através do atributo name:

```
<form name="f1">
<input type=button name="botao1" value="Botão 1">
<input type=text name="campoTexto" value="Texto Muito Velho">
</form>
```

Agora é possível ter acesso ao campo de textos em JavaScript usando nomes, em vez de índices de vetores:

```
texto = document.f1.campoTexto;
textoVelho = texto.value; // lendo a propriedade value...
texto.value = "Novo Texto";
```

O código do slide anterior também poderia ter sido escrito da forma, com os mesmos resultados:

```
textoVelho = document.f1.campoTexto.value;
document.f1.campoTexto.value = "Novo Texto";
```

A seguir veremos o exemplo para somar 2 números. O layout HTML terá a seguinte estrutura.

Somador JavaScript

Código JavaScript:

OBS: A função parseFloat converte uma string para um valor numérico com casa decimal.

```
<script language=JavaScript>
function soma() {
a = document.f1.val1.value;
b = document.f1.val2.value;
document.f1.val3.value = parseFloat(a) + parseFloat(b);
}
</script>
```

Código HTML:

```
<body>
<h1>Somador JavaScript</h1>
<form name="f1">
<input type=text name="val1" size="5"> +
<input type=text name="val2" size="5">
<input type=button value="somar" onclick="soma()">
<input type=text name="val3" size="5">
</form>
</body>
```

JavaScript possui várias funções e objetos nativos, que são procedimentos que permitem realizar tarefas úteis no dia-dia, como conversão de tipos, cálculos com datas, funções matemáticas, entre outras.

Todas recebem parâmetros com os dados sobre os quais devem operar. Podem ser chamadas de qualquer lugar. Por exemplo:

```
ano = parseInt("2010");
```

A função acima converte uma String para a sua representação númerica. Ignora qualquer coisa depois do ponto decimal ou depois de um caractere que não é número.

Se primeiro caractere não for número, retorna NaN (Not a Number).

A função parseFloat Converte uma String para a sua representação númerica, levando em consideração o ponto decimal. Ignora qualquer coisa depois do segundo ponto decimal ou depois de um caractere que não é número. Se primeiro caractere não for número ou ponto decimal, retorna NaN (Not a Number)

```
valor = parseFloat("2.98");
```

isNaN retorna true se o valor passado não é um número.

```
if (!isNaN(valor))
{
document.write('O valor informado não é um campo numérico');
}
```

O tipo String existe para dar suporte e permitir a invocação de métodos sobre cadeias de caracteres, representadas pelo tipo primitivo string. Pode-se criar um novo objeto String fazendo:

```
s = new String("string");
```

ou simplesmente

```
s = "string";
```

Objetos String possuem apenas uma propriedade: length, que pode ser obtida a partir de qualquer objeto string e contém o comprimento da cadeia de caracteres:

```
tamanho = s.length;
```

Os dois métodos a seguir realizam transformações no formato dos caracteres. São extremamente úteis em comparações e rotinas de validação. Retornam String.

Método Invocado	Retorno	Exemplo
toLowerCase()	texto (converte para caixa-baixa)	valor = valor.toLowerCase();
toUpperCase()	TEXTO (converte para caixa-alta)	valor = valor.toUpperCase ();

Os métodos seguintes realizam operações baseados nos caracteres individuais de uma string. Não afetam os strings originais. As transformações são retornadas.

Método Invocado	Retorno	Exemplo
charAt(*n)*	Retorna o caractere na posição *n.*	primeiraletra = valor. charAt(0);
indexOf("substring")	*Retorna um índice n referente à posição da* primeira ocorrência de "substring" na string s.	indice = valor. indexOf("s");
lastIndexOf("substring")	Retorna um índice n referente à posição da última ocorrência de "substring" na string s.	indice = valor. lastIndexOf("s");

Método Invocado	Retorno	Exemplo
split("delimitador")	Converte o string em um vetor de strings separando-os pelo "delimitador" especificado.	data = "Sexta-feira, 13 de Agosto de 1999"; sexta = data.split(",");
substring(inicio, fim)	Extrai uma substring de uma string s. •inicio é um valor entre 0 e s.length-1. •fim é um valor entre 1 e s.length. O caractere na posição inicio é incluído na string e o caractere na posição fim não é incluído. A string resultante contém caracteres de inicio a fim -1.	pedaco= valor. substring(0,2);

O objeto Math contém um conjunto de funções matemáticas. Para ter acesso a suas funções e constantes, deve-se usar a sintaxe:

```
Math.função();
```

Funções			
`acos(x)`	cosseno^{-1}	`abs(x)`	absoluto
`asin(x)`	seno^{-1}	`max(a, b)`	máximo
`atan(x)`	tangente^{-1}	`min(a, b)`	mínimo
`atan2(x, y)`	retorna o ângulo θ de um ponto $(x;y)$	`pow(x, y)`	x^y
		`sin(x)`	seno
`ceil(x)`	arredonda para cima (3.2 e 3.8 → 4)	`round(x)`	arredonda (3.49 → 3 e 3.5 → 4)
`cos(x)`	cosseno	`tan(x)`	tangente
`exp(x)`	e^x	`sqrt(x)`	raiz quadrada
`floor(x)`	arredonda para baixo (3.2 e 3.8 → 3)	`log(x)`	logarítmo natural
`random()`	retorna um número pseudo-aleatório entre 0 e 1.		

O tipo Date é um tipo de objeto usado para representar datas. Para criar data que represente a data e hora atuais, chame-o usando new, da forma.

```
aquiAgora = new Date();
```

Para utilizar as informações de um Date, invoca-se os seus métodos sobre o objeto criado. Métodos podem ser invocados a partir de um objeto Date como no exemplo a seguir:

```
dia = umDia.getDay();
hora = umDia.getHours();
ano = umDia.getYear();
document.writeln("Horário de Greenwich: " +
umDia.toGMTString());
```

A tabela a seguir relaciona os métodos dos objetos do tipo Date, os tipos de retorno (se houver) e suas ações.

Método	Ação
`getDate()`	Retorna *Number*. Recupera o dia do mês (1 a 31)
`getDay()`	*Number*. Recupera o dia da semana (0 a 6)
`getHours()`	*Number*. Recupera a hora (0 a 23)
`getMinutes()`	*Number*. Recupera o minuto (0 a 59)
`getMonth()`	*Number*. Recupera o mês (0 a 11)
`getSeconds()`	*Number*. Recupera o segundo (0 a 59)
`getTime()`	*Number*. Recupera a representação em milissegundos desde 1-1-1970 0:0:0 GMT
`getTimezoneOffset()`	*Number*. Recupera a diferença em minutos entre a data no fuso horário local e GMT (não afeta o objeto no qual atua)
`getYear()`	*Number*. Recupera ano menos 1900 (1997 → 97)

Método	Ação
`setDate(dia_do_mês)`	Acerta o dia do mês (1 a 31)
`setHours(hora)`	Acerta a hora (0 a 23)
`setMinutes(minuto)`	Acerta o minuto (0-59)
`setMonth(mês)`	Acerta o mês (0-11)
`setSeconds()`	Acerta o segundo (0-59)
`setTime()`	Acerta a hora em milissegundos desde 1-1-1970 0:0:0 GMT
`setYear()`	Acerta o ano (ano – 1900)
`toGMTString()`	*String*. Converte uma data em uma representação GMT
`toLocaleString()`	*String*. Converte a data na representação local do sistema

8.4. Exemplos práticos

Neste tópico veremos alguns exemplos práticos em JavaScript:

- Abrir uma janela popup
- Janela de alerta após o carregamento /saída da página
- Mensagem de Confirmação
- Cor de fundo
- Janela pop-up programada
- Hora certa na barra de status
- Input text com texto padrão e valor em branco quando usuário clicar
- Verificando caracteres ao digitar em uma input text

ABRIR JANELA DO NAVEGADOR

A JANELA DO BROWSER é manipulável de várias formas através da linguagem JavaScript. Pode-se alterar dinamicamente várias de suas características como tamanho, aparência e posição. Para abrir uma janela usamos o método window.open(). Veja exemplo:

```
window.open('URL','titulo da
página','top=valor,left=valor,width=valor,height=valor);
```

```
<body
onload="window.open('teste.html','pagina','top=400,left=400,widt
h=50,height=50');" >
```

JANELA DE ALERTA APÓS CARREGAMENTO OU SAÍDA DO NAVEGADOR

Podemos inserir uma mensagem de alerta ao carregar uma página ou quando o usuário sair dela. Usamos os métodos onunload ou onload da tag body.

```
<body
onload="alert('Bem vindo');"
onunload="alert('Obrigado pela visita, volte sempre!');">
```

MENSGEM DE CONFIRMAÇÃO

Para exibir uma mensagem de confirmação usamos a função confirm("Mensagem").

Veja exemplo:

```
function abrirJanela(){
    if (confirm("Tem certeza que deseja abrir esta página?"))
     {
       window.open("teste.html");
     }
     }
```

```
<body
onload="alert('Bem vindo');"
```

```
onunload="alert('Obrigado pela visita, volte sempre!');">
```

Podemos permitir que o usuário escolha a cor de fonte do site. Para isso iremos criar uma página com três inputs do tipo radio. Em cada input usaremos o método onclick com o código javascript abaixo:

COR DE FUNDO

```
document.bgColor='cor'
```

```
Azul <input TYPE="radio" onClick="document.bgColor='blue'">
Vermelho <input TYPE="radio" onClick="document.bgColor='red'">
Amarelo <input TYPE="radio" onClick="document.bgColor='yellow'">
```

JANELA POP-UP PROGRAMADA

A função setTimeout("instruções",atraso) Executa uma ou mais instruções JavaScript após um período de atraso em milissegundos.

A função retorna um número de identificação que pode ser passado como argumento do método clearTimeout() para executar a operação imediatamente, ignorando o tempo que falta.

A função clearTimeout(id) cancela a temporização de uma operação setTimeout() cujo número de identificação foi passado como parâmetro, e faz com que as instruções do setTimeout() sejam interpretadas e executadas imediatamente.

No exemplo a seguir é criada uma função que abre uma janela e fecha a mesma em 3 segundos.

```
function abrirJanela(){
        janela = window.open("teste.html");
    setTimeout('janela.close();', 3000);
  }
```

Para executa-la, usaremos o método onclick da tag body.

```
<body onload="abrirJanela();">
```

HORA CERTA NA BARRA DE STATUS DO NAVEGADOR

Código JavaScript

```
function iniciarrelogio(){
       setTimeout('mostrarhora();', 1000);}

function mostrarhora(){
      var hora = new Date();

        h = hora.getHours();
      m = hora.getMinutes();
      s = hora.getSeconds();
      window.status = "Olá bem vindo! A hora certa é:" + h + ":"
+ m + ":" + s;

      setTimeout('mostrarhora();', 1000);}
```

No HTML:

```
<body onload="iniciarrelogio();" >
```

INPUT TEXT COM TEXTO PADRÃO E VALOR EM BRANCO QUANDO USUÁRIO CLICAR

Podemos aplicar um efeito simples mas bastante útil para passar informações ao usuário quando usarmos formulário.

Neste exemplo usaremos um input text, com um texto padrão quando ele estiver inativo e em branco quando o usuário clicar no campo.

```
<label>Nome: </label>
<input type="text" name="nome" onblur="this.value='Digite o nome
completo';" onclick="this.value='';"/>
```

VERIFICANDO CARACTERES AO DIGITAR EM UMA INPUT TEXT

Podemos desenvolver código Javascript para realizar validações no momento que o usuário digitar alguma caractere em uma input text. Para isso, usaremos o método onkeydown.

Para verificar qual o caractere digitado, teremos que recorrer a tabela ASC para verificar o valor da tecla e checar a condição. OBS: Cada tecla do teclado tem um código correspondente.

No exemplo abaixo, a validação é feita para permitir que apenas caracteres numéricos sejam aceitos na entrada de dados.

```
<label>Nome: </label>
<input type="text" name="nome"
onkeydown="if (event.keyCode< 48 || event.keyCode >
57){alert('Você só pode digitar números neste campo.'); return
false;}"
```