Aula 18: Crítica de Formulários

Nesta aula você verá os diversos objetos relacionados aos formulários. Veremos como é possível usar os métodos e propriedades dos diversos elementos de formulários e seus eventos para verificar a correção dos dados digitados pelo usuário.

Objetivos:
Nesta aula você aprenderá:
- as propriedades e métodos do objeto `form`;
- os vetores de elementos do formulário;
- como fazer crítica de campo e
- crítica de formulário.

Pré-requisitos:
A aula 7 do primeiro módulo e as aulas 12 a 17 do segundo são importantes para esta aula.

1. Formulários como Objeto de JavaScript

Uma das principais aplicações de JavaScript é a possibilidade de criticar dados fornecidos pelo usuário através de formulários HTML. O conteúdo dos formulários pode ser acessado pelo script através de instâncias do objeto `Form`, guardadas no array `forms` do objeto `Document`. Um formulário também pode ser acessado pelo nome definido no atributo `name` da tag `form`.

```
<FORM NAME="meuform">
</FORM>//definiu o formulario
....
document.meuform   //acesso pelo nome
document.forms[0]  //acesso pelo array
```

O objeto `Form` tem as propriedades e métodos descritos nas tabelas 18.1 e 18.2. Além das formas de tratamento de eventos que recebem por herança dos elementos HTML, o objeto `Form` também responde aos eventos da tabela 18.3.

Tabela 18.1 - Propriedades do objeto `Form`

Propriedade	Descrição
`action`	*String* especificando o atributo `action` do formulário (URL de submissão).
`elements[]`	*Array* dos elementos de entrada: `text, radio, button, checkbox` etc.
`encoding`	*String* especificando o método de codificação usado para os dados do *form*.
`length`	Número de elementos do formulário.
`method`	*String* especificando o método de envio.
`name`	O nome do formulário.
`target`	*String* com o nome do *frame* ou *window* no qual os resultados de submissão do *form* devem ser mostrados.

Tabela 18.2 - Métodos do objeto `Form`

Método	Descrição
`submit()`	Envia o formulário ao servidor.
`reset()`	Redefine cada campo com os valores *default*.

Tabela 18.3 - Eventos do objeto `Form`

Evento	Descrição
`onSubmit`	Permite associar uma função ao *submit* do formulário.
`onReset`	Permite associar uma função ao *reset* do formulário.

É possível associar funções aos eventos `submit` ou `reset` do formulário, que serão ativadas quando os respectivos botões forem selecionados. Através da captura do evento `submit` é possível fazer a verificação final nos campos do formulário, evitando o envio se ainda houver algum erro. Se alguma destas funções retornar `FALSE` a ação correspondente é cancelada. No exemplo a seguir o navegador pede uma confirmação se o usuário quer mesmo limpar um formulário:

```
onReset=
"return confirm('Quer mesmo apagar?')"
```

2. Elementos de um Formulário

O objeto `Form`, como já mencionado, contém um array, onde são armazenados os elementos de interação do formulário. Um elemento pode ser acessado pelo array `elements` ou pelo nome definido no HTML. Por exemplo, dado o formulário:

```
<FORM NAME="meuform">
<INPUT TYPE=TEXT NAME="endereco">
</FORM>
```

As duas formas a seguir são equivalentes para referenciar o elemento área de entrada de texto:

```
document.meuform.endereco
document.forms[0].elements[0]
```

É possível determinar o número de elementos de um formulário através da propriedade `length` de `Form`. Como os elementos estão armazenados em um array (que também tem propriedade `length`), as duas formas abaixo são equivalentes:

```
document.forms[0].elements.length
document.forms[0].length
```

Convém lembrar, porém, que `forms` também é um *array* e, portanto, se utilizarmos `length` sem indicar um índice estamos na verdade acessando o número de formulários do documento, como no exemplo abaixo:

```
document.forms.length
```

Cada posição do vetor `elements` pode conter qualquer um dos elementos

de interação que vimos em HTML, que são:

- Campos de texto: `text, password, hidden`
- Áreas de texto: `textarea`
- Botões: `button, reset, submit`
- Caixas de seleção: `checkbox`
- Botões de opção: `radio`
- Listas drop-down: `select`

2.1. Campo de Texto

As tabelas 18.4 a 18.6 apresentam as propriedades, métodos e eventos dos **campos de texto**.

Tabela 18.4 - Propriedades de `text, password e hidden`

Propriedade	Descrição
`name`	Nome associado ao elemento.
`value`	Valor digitado pelo usuário.
`defaultValue`	Valor a ser exibido no elemento após um *reset*.

Tabela 18.5 - Métodos de `text, password e hidden`

Método	Descrição
`focus( )`	Coloca o elemento ativo (em foco).
`blur()`	Coloca o elemento inativo (remove o foco).
`select()`	Seleciona o texto no campo.

Tabela 18.6 - Eventos de `text, password e hidden`

Evento	Descrição
`onFocus`	Ocorre quando o campo recebe o foco.
`onBlur`	Ocorre quando o campo perde o foco.
`onChange`	Ocorre quando o valor do campo muda.
`onSelect`	Ocorre quando o usuário seleciona o texto do campo.

Nas linhas que seguem apresenta-se um exemplo de utilização de alguns dos elementos das tabelas anteriores. A figura que acompanha o exercício 1 no final desta aula pode lhe dar uma idéia do que ocorre quando este exemplo é visualizado em um *browser*.

```
<html>
<head>
<title>Critica de campo</title>
<script language="javascript">
<!--
//Retorna se o valor for numerico
function EhDigito(car)
{
  return((car >= "0") && (car <= "9"));
}
//Verifica se o telefone está correto
function VerificaTelefone(campo)
{
  var telef = campo.value;
  for (i = 0; i < telef.length; i++)
```

```
    if (! EhDigito (telef.charAt(i)))
    {
     alert("Caracter "+telef.charAt(i)+
           " inválido:deve ser numero!");
     campo.focus();
     return false;
     }
    return true;
}
//-->
</script>
</head>

<body bgcolor=navy text=yellow>
<form>
<H2>Critica de campo</H2>
  <TABLE>
  <TR>
  <TD>Nome:</TD>
  <TD><INPUT TYPE="text" NAME="nome"  VALUE=""></TD>
  </TR>
  <TR>
  <TD>Telefone:</TD>
  <TD><INPUT TYPE="text" NAME="telefone" VALUE=""
      onChange= "VerificaTelefone(this)"></TD>
  </TR>
  </TABLE>
<P>
  <input type="button" value="Enviar">
  <input type="reset"  value="Apagar">
</P>

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

2.2. Botões tipo: `button, reset e submit`

As tabelas 18.7 e 18.8 apresentam as propriedades e métodos dos **botões tipo:** `button, reset e submit`.

Tabela 18.7 - Propriedades dos botões tipo: `button, reset e submit`

Propriedade	Descrição
`name`	Nome associado ao elemento.
`value`	Valor exibido no botão.

Tabela 18.8 - Eventos dos botões tipo: `button, reset e submit`

Evento	Descrição
`onFocus`	Ocorre quando o botão recebe o foco.
`onBlur`	Ocorre quando o botão perde o foco.
`onClick`	Ocorre quando o botão é selecionado com o mouse.
`onMouseDown`	Ocorre quando o botão do mouse é pressionado.
`onMouseUp`	Ocorre quando o botão do mouse é levantado.

2.3. Botões tipo `checkbox`

As tabelas 18.9 a 18.10 apresentam as propriedades e métodos dos `checkbox`.

Tabela 18.9 - Propriedades dos botões `checkbox`

Propriedade	Descrição
`name`	Nome associado ao elemento.
`checked`	Booleano que indica se a checkbox está selecionada.
`defaultChecked`	Se a checkbox estará selecionada após um reset.

Tabela 18.10 - Métodos dos botões `checkbox`

Método	Descrição
`onFocus`	Ocorre quando o campo recebe o foco.
`onBlur`	Ocorre quando o campo perde o foco.
`onClick`	Ocorre quando a checkbox é selecionada com o mouse.

As linhas que seguem exemplificam o uso de alguns dos elementos destas tabelas, produzindo a figura abaixo.

Figura 18.1 - Exemplo de uso de checkbox


```
<HTML>
<HEAD>
<TITLE>Exemplo de checkbox</TITLE>
<SCRIPT>
function clicou(campo)
{
 if (campo.checked)
   alert("O campo está selecionado");
 else
   alert("Campo desmarcado !");
}
</SCRIPT>
</HEAD>
<BODY BGCOLOR=beige>
<H1>Exemplo de checkbox</H1>
<FORM>
   <INPUT TYPE=checkbox onClick=
     "clicou(this)">Marque aqui!
</FORM>
</BODY>
</HTML>
```

2.4. Botões tipo `radio`

Um grupo de **botões de radio** (exclusivos) é formado por vários elementos criados com a tag `<INPUT TYPE=RADIO>` e o mesmo valor para o atributo `NAME`. Estes elementos são tratados em conjunto, pois o seu comportamento depende do grupo (apenas um dos botões do grupo pode estar selecionado em um determinado instante de tempo). Como neste objeto é sempre obrigatório a definição do atributo `NAME`, o grupo de botões de radio é manipulado através de um array com este nome no objeto `Form`. Assim, se no formulário `form1` tivermos um grupo de botões de radio chamado `radio1`, cada exemplo a seguir mostra a sintaxe correta para:

- Saber o número de botões agrupados
  ```
  document.form1.radio1.length
  ```
- Verificar se o primeiro botão está selecionado
  ```
  if (document.form1.radio1[0].checked)
  ```
- Testar o atributo `value` do segundo botão

```
if (document.form1.radio1[1].value==...)
```

As propriedades deste objeto (tabela 18.11) são muito parecidas com as do objeto `checkbox.` A tabela 18.12 mostra os eventos deste tipo de botão.

Tabela 18.11 - Propriedades dos botões de `radio`

Propriedade	Descrição
`name`	Nome associado ao elemento.
`checked`	Booleano que indica se o botão de radio está selecionado.
`defaultChecked`	Se o botão estará selecionado após um *reset*.

Tabela 18.12 - Eventos dos botões de `radio`

Evento	Descrição
`onFocus`	Ocorre quando o botão recebe o foco.
`onBlur`	Ocorre quando o botão perde o foco.
`onClick`	Ocorre quando o botão é selecionado com o mouse.

As linhas de código abaixo exemplificam alguns dos elementos destas tabelas, que irão gerar uma figura muito parecida com a mostrada no exercício 3 no final da aula.

```
<HTML>
<HEAD>
<TITLE>Exemplo de Radio</TITLE>
</HEAD>
<SCRIPT LANGUAGE="javascript">
function convertField(field)
{
 if(document.form1.conversion[0].checked)
   field.value=field.value.toUpperCase()
}
function convertAllFields(caseChange)
```

```
{
 with (document.form1)
   if (caseChange=="upper")
   {
     lastName.value=
          lastName.value.toUpperCase();
     firstName.value=
          firstName.value.toUpperCase();
   }
}
</SCRIPT>
<BODY BGCOLOR="lightSteelBlue">
<H1>Exemplo de Radio</H1>
  <FORM NAME="form1">
   <TABLE>
    <TR>
    <TH ALIGN=LEFT>:&Uacute;ltimo nome:</TH>
    <TD>
    <INPUT         TYPE="text"         NAME="lastName"
      onChange="convertField(this)">
    </TD>
    </TR>
    <TR>
    <TH ALIGN=LEFT>Primeiro nome:</TH>
    <TD>
    <INPUT        TYPE="text"        NAME="firstName"
      onChange="convertField(this)">
    </TD>
    </TR>
   </TABLE>
      <P>
      <B>Valores convertidos para:</B>
      <BR>
      <INPUT TYPE="radio" NAME="conversion"
          VALUE="upper" onClick="if (this.checked)
          convertAllFields('upper');">
          Mai&uacute;sculas
  </FORM>
 </BODY>
</HTML>
```

2.5. Listas de Seleção

As listas de seleção (*drop-down*) são um tipo de elemento de interação que envolve dois tipos de objeto: o objeto `select`, que representa a lista, e o objeto `option`, que mostra cada uma das opções. As propriedades do objeto `select` e seus eventos podem ser observadas nas tabelas 18.13 e 18.15, enquanto que as propriedades de `option` se encontram na tabela 18.14.

Tabela 18.13 - Propriedades de `select`

Propriedade	Descrição
`name`	Nome associado ao elemento.
`length`	Número de opções da lista.
`options`	Array com as opções.
`selectedIndex`	Índice da opção atualmente selecionada. Se for uma lista de seleção múltipla é o índice da primeira seleção.

Tabela 18.14 - Propriedades de `option`

Propriedade	Descrição
`index`	Índice no *array*.
`defaultSelected`	Se a opção estará selecionada após um reset.
`selected`	Booleano que indica se a opção está selecionada.
`text`	Texto exibido na opção.
`value`	Valor associado a opção.

Tabela 18.15 - Eventos de `select`

Evento	Descrição
`onFocus`	Ocorre quando a lista recebe o foco.
`onBlur`	Ocorre quando a lista perde o foco.
`onClick`	Ocorre quando a lista é selecionada com o mouse.

A linguagem JavaScript permite que o `select` seja modificado após a página ter sido carregada. É possível:

- modificar o texto que está sendo exibido

```
document.f1.s1.options[2].text="novo";
```

- modificar o valor atribuído à opção

```
document.f1.s1.options[1].value="v1";
```

- criar uma nova opção

```
document.f1.s1.options[8] = new option ("texto" ,
            "valor");
```

- remover uma opção

```
document.f1.s1.options[4] = null;
```

Para criar uma nova opção é preciso criar um novo objeto `option` através do operador `new`. O construtor de `option` pode receber até 4 parâmetros (opcionais):

- texto que será exibido
- valor da opção
- `defaultSelect`
- indicação se estiver selecionada

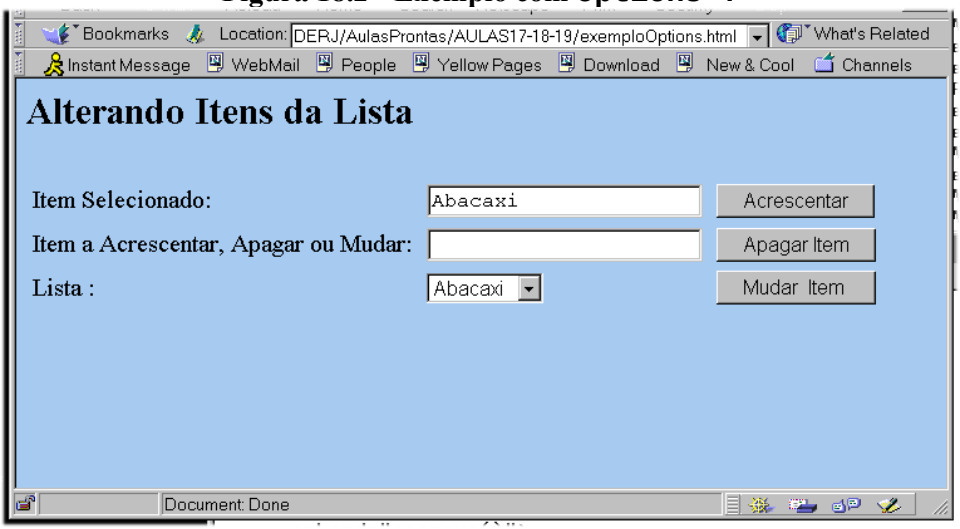
A remoção de uma opção é feita atribuindo à posição respectiva do *array* `options` o valor `null`. Qualquer opção pode ser removida, inclusive aquela que está atualmente selecionada. As linhas de código a seguir ajudam a entender isso:

```
<HTML>
<HEAD>
<TITLE>Alterando Itens</TITLE>
<SCRIPT LANGUAGE="javascript">
function inclui()
{
  with (document.formulario)
  {
    if (novo.value == "")
    {
     alert("Novo item ruim");
     return;
   }
   var nop = lista.length;
   lista.options[nop]=new
                   Option(novo.value);
   }
}
function escreve()
{
   with (document.formulario)
   {
     var ind = lista.selectedIndex;
     selecionado.value =
          lista.options[ind].text
   }
}
function remove()
{
   with (document.formulario)
   {
      var ind = lista.selectedIndex;
      lista.options[ind] = null;
   }
}
function muda()
{
  with (document.formulario)
  {
    if (novo.value == "")
    {
      alert("Novo item ruim!");
      return;
    }
    var ind = lista.selectedIndex;
    lista.options[ind].text = novo.value;
  }
}
</SCRIPT>
</HEAD>
<BODY BGCOLOR=lightsteelblue
      onLoad="escreve()">
<H2>Alterando Itens da Lista</H>
 <FORM NAME="formulario">
 <TABLE>
 <TR>
 <TD>Item Selecionado:
 <TD><INPUT TYPE=text NAME=selecionado
       onFocus="this.forms.novo.focus()">
```

```
 <TD><INPUT TYPE=BUTTON
            VALUE="Acrescentar"
            onClick="inclui()">
 <TR>
 <TD>Item a Acrescentar, Apagar ou Mudar:
 <TD><INPUT TYPE=text NAME=novo>
 <TD><INPUT TYPE=BUTTON
            VALUE="Apagar Item"
            onClick="remove()">
 <TR>
 <TD>Lista :
 <TD>
 <SELECT NAME=lista onchange="escreve()">
 <OPTION checked>Abacaxi
 <OPTION>Banana
 <OPTION>Laranja
 <OPTION>Goiaba
 </SELECT>
 <TD><INPUT TYPE=BUTTON
            VALUE="Mudar  Item "
            onClick="muda()">
 </TABLE>
</FORM>
</BODY>
</HTML>
```

As linhas de código do exemplo acima produzem a página mostrada na figura 18.2 que segue.

Figura 18.2 - Exemplo com `options` .



2.6. Área de Texto

Área de texto é um elemento de interação bastante versátil e pode ser utilizado para exibição de mensagens geradas durante a execução do JavaScript. Escrever mensagens numa área de texto tem uma vantagem em relação ao `document.write`, pois, mesmo que a página já esteja completa, é possível acrescentar texto à página sem que se perca o conteúdo anterior (um `write` após o carregamento da página faz com que seja criado um novo documento, apagando o anterior). As propriedades, métodos e os eventos deste objeto se encontram nas tabelas 18.16 a 18.18.

Tabela 18.16 - Propriedades das áreas de texto

Propriedade	Descrição
`name`	Nome associado ao elemento.
`value`	Valor digitado pelo usuário.
`defaultValue`	Valor a ser exibido no elemento após um *reset*.

Tabela 18.17 - Métodos das áreas de texto

Método	Descrição
`focus( )`	Coloca o elemento ativo (em foco).
`blur()`	Coloca o elemento inativo (remove o foco).
`select()`	Seleciona o texto no campo.

Tabela 18.18 - Eventos das áreas de texto

Evento	Descrição
`onFocus`	Ocorre quando o campo recebe o foco.
`onBlur`	Ocorre quando o campo perde o foco.
`onChange`	Ocorre quando o valor do campo muda.
`onSelect`	Ocorre quando o usuário seleciona o texto do campo.
`onKeyDown`	Ocorre quando uma tecla é pressionada.
`onKeyUp`	Ocorre quando uma tecla é solta.
`onKeyPress`	Pressionar e soltar uma tecla.

3. Crítica do Formulário

Numa página com um formulário, a utilização de JavaScript permite auxiliar o usuário e evitar que ele **cometa erros de preenchimento**. Este procedimento é chamado de **crítica do formulário.**

A crítica do formulário pode ser realizada após a **modificação do valor de algum elemento** de interação (para isto utiliza-se, normalmente, o evento `onChange`) ou após o usuário ter pedido ao navegador para **enviar o formulário** (para isso utiliza-se o evento `onSubmit` do formulário ou o evento `onClick` do botão de `submit`).

Normalmente, é necessário fazer os dois tipos de verificação: **a crítica do campo** permite detectar erros imediatamente após o usuário tê-los cometido e a **crítica do formulário** permite detectar erros resultantes da interdependência de dois ou mais campos.

O exemplo a seguir ilustra como é feita a crítica de um **campo do formulário**. A função do exemplo testa se um campo de texto (no caso um endereço) foi preenchido e informa ao usuário que o campo é obrigatório, caso ele não tenha sido definido:

```
function critica(campo)
{
  if (campo.value.length == 0)
```

```
    alert('Campo ' + campo.name +
          ' nao preenchido.');
}
...
<form ...>
<input type=text name=endereco
       onChange="critica(this)">
...
</form>
```

O exemplo a seguir ilustra a **crítica final** de um formulário. Este exemplo mostra um formulário onde deve-se digitar duas datas, com campos para dia, mês e ano. Antes de enviar o formulário é feita a verificação se a data de saída é posterior a data de entrada. Isso só pode ser feito quando o formulário já está pronto, pois não se pode forçar a ordem do usuário definir o valor dos campos:

```
function criticar()
{
  with (document.forms[0])
  {
               var dataent = new Date( ano_ent.value,
                                        mes_ent.value,
                                       dia_ent.value);
    var datasai = new Date(ano_sai.value,
                                        mes_sai.value,
                                       dia_sai.value);
    if(dataent.getTime()>=datasai.getTime())
    {
                  alert("Saida deve ser posterior" +
                   " a entrada !");
       return false;
    }
  }
  return true;
}
. . .
<form action="/cgi-bin/x"
      onSubmit="return criticar()">
. . .
</form>
```

Alguns tipos de crítica podem servir inclusive para modificar o valor de um campo de forma a ficar coerente com o valor de outro. O exemplo a seguir mostra como a seleção da nacionalidade como estrangeira causa a "limpeza" do campo naturalidade:

```
<form name="meuform" ...>
  <P>Nacionalidade:<br>
  <input type="radio" value="1" name="nacionalidade">
    Brasileiro<br>
  <input type="radio" value="2" name="nacionalidade"
    onClick="document.meuform.naturalidade.value = "">
    Estrangeiro
  <P>Naturalidade:
```

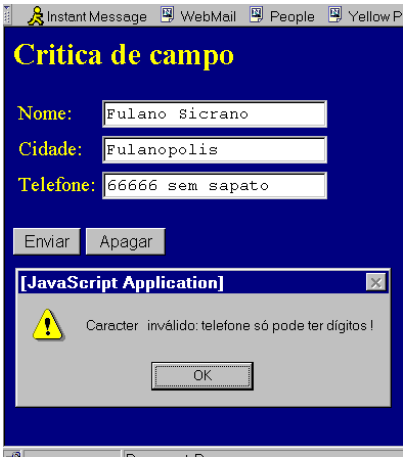
```
  <input type="text" name="naturalidade"
    onChange="validaNaturalidade(this)">
  <br>
...
</form>
```

Exercícios:

1. Digite em seu computador o código que foi mostrado na seção 2.1 desta aula.
Primeiro faça a página ser carregada na forma original. Depois modifique o código para:

- incluir um campo de texto para ser fornecido à cidade (veja figura 18.3 que segue);
- usando a idéia de verificação do telefone, faça uma função para verificar se só letras foram digitadas;
- use esta função para verificar o nome e também o novo campo que você criou.

Figura 18.3 - Tela relacionada ao exercício 1



2. Amplie a idéia de verificação de `checkbox`, que gerou a figura mostrada na seção 2.3, incluindo no exercício anterior 3 destes botões, com alguma sugestão para verificação (Por exemplo: "permite que divulguemos seu telefone"; "permite divulgação do seu nome", "não permite qualquer divulgação"). Verifique se fazem sentido as opções do usuário (por exemplo: a última exclui as duas primeiras no exemplo de sugestão).

3. Utilize as linhas de código do exemplo de botões de radio da seção 2.4 para entender a forma de uso da propriedade `checked`. Após entender bem o exemplo, amplie as opções para também passar as letras para "Minúsculas" ou deixá-las "Sem conversão" (inalteradas) , como se mostra na figura 18.4 que segue.

Figura 18.4 - Tela relacionada com o exercício 3.

4. Utilizando a idéia do exemplo da seção 2.5, inclua no formulário do exercício 1 uma lista com alguns estados do país, ao lado da cidade. Inclua também a possibilidade de ter um campo para o usuário incluir um novo estado caso queira.

5. Após entender bem a seção 3, inclua no exercício 4 uma forma de verificação final do formulário. Pode ser, por exemplo, a verificação de que o usuário não esqueceu de preencher os campos de nome e endereço.

Resumo:

Nesta aula você aprendeu a usar os objetos relacionados ao formulário para verificar itens ligados ao seu preenchimento pelo usuário. Estamos quase terminando este curso! Na próxima aula veremos como manipular os frames e janelas. Até lá!

Auto-avaliação:

Esta aula, como a anterior, é bastante complexa, não? Mas, como você se saiu nos exercícios? Retorne aos pontos onde sentiu mais dificuldade, e não se preocupe muito se ainda não estiver bem seguro mesmo depois disto. Devemos ser sinceros e lhe dizer que o assunto é bem amplo mesmo. Afinal, já estamos acabando o curso e só a prática lhe dará a sensação de completa segurança...