

9 Interação

Neste capítulo você vai aprender a utilizar elementos interativos tais como botões, caixas de textos, listas com opções de escolha e figuras interativas em seu documento. Vai aprender também como capturar e gerenciar informações de eventos, tais como movimentos e cliques do mouse e teclas digitadas.

input type="button"

É possível criar documentos interativos adicionando funções gerenciadoras de eventos aos elementos de um documento. No exemplo a seguir, o botão muda de cor quando o ponteiro do mouse está sobre ele e a função `sorteia()` é acionada quando for clicado:

Listagem:

```
<script>
function sorteia() {
    var num = parseInt(1 + Math.random()*10);
    alert("Você deve resolver a questão número " + num);
}
</script>

<p style='text-align:center'>
Clique no botão abaixo para receber sua tarefa:<br>
<input
    type='button'
    value='BOA SORTE'
    style = "background-color:lightgray"
    onClick="sorteia()"
    onMouseOver="this.style.backgroundColor = 'lightgreen'"
    onMouseOut="this.style.backgroundColor = 'lightgray'"
/>
</p>
```

Resultado:

Clique no botão abaixo para receber sua tarefa:

BOA SORTE

O elemento utiliza três gerenciadores de eventos para controlar o que acontece quando o usuário clica sobre o botão (`onClick`), quando o mouse passa sobre o botão (`onmouseover`) e quando o mouse deixa o botão (`onmouseout`). No primeiro caso, ao gerenciador de eventos foi atribuída a tarefa de chamar a função `sorteia()`. Nos outros dois, uma instrução JavaScript foi diretamente atribuída aos gerenciadores de eventos. A palavra `this` (este) nessas instruções indica que a operação deve ser realizada sobre o elemento que gerou o evento, que é o próprio botão.

Note que no atributo `style` a propriedade que define a cor de fundo é escrita com um hífen entre `background` e `color`, e não importa se letras maiúsculas ou minúsculas são utilizadas. Já na instrução JavaScript, a propriedade é escrita sem o hífen e as maiúsculas e minúsculas devem ser estritamente respeitadas. Isso porque o `background-color` está no contexto do HTML, no qual um hífen é somente um caracter, enquanto o `backgroundColor` está no contexto do JavaScript, em que o hífen é, na realidade, um sinal de subtração.

Sua vez... (9-1)

Modifique o exemplo anterior de modo que não aconteça nada quando o botão for clicado, mas sim quando o mouse passar sobre ele e, uma vez sorteada a questão, o botão fique permanentemente vermelho.

São vários os gerenciadores de eventos disponíveis, e entre os mais usuais estão:

<code>onClick</code>	Clique com o mouse
<code>ondblclick</code>	Clique duplo com o mouse
<code>onchange</code>	Conteúdo de elemento modificado
<code>onblur</code>	Elemento perde o foco
<code>onfocus</code>	Elemento recebe o foco
<code>onmousemove</code>	O mouse anda
<code>onmousedown</code>	Botão pressionado
<code>onmouseup</code>	Botão levantado
<code>onmouseover</code>	Mouse sobre o elemento
<code>onmouseout</code>	Mouse sai do elemento
<code>onload</code>	Um documento ou imagem é carregada
<code>onmouseout</code>	Mouse sai do elemento
<code>onkeydown</code>	Uma tecla é pressionada
<code>onkeypress</code>	Uma tecla é pressionada e mantida assim
<code>onkeyup</code>	Uma tecla é liberada
<code>onresize</code>	Uma janela é redimensionada

onSelect Um texto é selecionado

Em alguns casos, quando um destes gerenciadores de evento é chamado, é importante saber em que condições isso aconteceu, por exemplo, qual a posição do mouse, qual o botão acionado ou, no caso do teclado, que tecla adicional (CTRL, ALT etc.) estava simultaneamente pressionada.

altKey	Se a tecla "ALT" estava pressionada
ctrlKey	Se a tecla "CTRL" estava pressionada
shiftKey	Se a tecla "SHIFT" estava pressionada
button	Que botão do mouse foi pressionado
clientX	Coordenada horizontal do mouse
clientY	Coordenada vertical do mouse

O exemplo anterior pode ser modificado para de modo a apresentar uma mensagem diferente para cada tecla extra pressionada:

Listagem:

```
<script>
function questao(event) {
    var num = parseInt(1 + Math.random()*10);
    if (event.altKey) alert("ALT " + num );
    else if (event.ctrlKey) alert("CTRL " + (num+10) );
    else if (event.shiftKey) alert("SHIFT " + (num+20) );
    else alert(num);
}
</script>

<p>
Clique no botão abaixo para receber sua tarefa.
São três grupos de questões:
<ul>
<li>1-10: clique no botão com a tecla "ALT" pressionada
<li>11-20: clique no botão com a tecla "CTRL" pressionada
<li>21-30: clique no botão com a tecla "SHIFT" pressionada
</ul>
</p>

<p style='text-align:center'>
<input type='button' value='BOA SORTE' onClick="questao(event)">
</p>
```

Resultado:

Clique no botão abaixo para receber sua tarefa. São três grupos de questões:

- 1-10: clique no botão com a tecla "ALT" pressionada
- 11-20: clique no botão com a tecla "CTRL" pressionada
- 21-30: clique no botão com a tecla "SHIFT" pressionada

BOA SORTE

Sua vez... (9-2)

Modifique o exemplo anterior de modo a fazer com que o botão fique vermelho, verde ou azul quando as telcas ALT, CTRL e SHIFT estiverem pressionadas, respectivamente, quando o botão for clicado.

`input type="text"`

Outro importante elemento de interatividade em um documento HTML é o `input` do tipo `text`, conhecido como *caixa de texto*. No exemplo a seguir, é colocada no documento uma caixa de texto que solicita ao usuário que digite o seu nome e, a seguir, coloca-o todo em letras maiúsculas.

Listagem:

```
<script>
function valida() {
    var ref = document.getElementById('nome');
    ref.value = (ref.value).toUpperCase();
}
</script>

<p style='text-align:center'>
Digite o seu nome e pressione ENTER
</p>

<input type='text' size='20' id='nome'
value='<digite seu nome>' onChange='valida()' >
</p>
```

Resultado:

Digite o seu nome e pressione ENTER

<digite seu nome>

Ao ser detetada uma mudança (`onChange`) no conteúdo da caixa de texto, a função `valida()` é chamada. Esta função usa o método `getElementById()` para buscar no documento um elemento com a identidade (`id`) "nome" e coloca uma referência a ele na variável `ref`. Esta referência é essencialmente o "endereço" da caixa de texto dentro do documento (cada elemento de um documento HTML tem um endereço único que pode ser referenciado). Na linha seguinte, o atributo `value`, que contém o conteúdo da caixa, é transformado para letras maiúsculas e o valor resultante da transformação atribuído novamente ao atributo `value` do elemento.

As instruções na função `valida()` fazem algo parecido com o que faz a instrução `x = x + 1`, que pega o valor da variável `x`, modifica-o e coloca o resultado na própria variável `x`).

Sua vez... (9-3)

Modifique o exemplo anterior de modo a inserir outra caixa de texto com outro `id` e fazer com que a função `valida()` transfira o texto digitado na primeira caixa para a outra.

Combinações de caixas de texto com botões permitem a construção de uma calculadora elementar:

Listagem:

```
<script>
function operacao(opt) {
  var op1 = parseFloat(document.getElementById('op1').value);
  var op2 = parseFloat(document.getElementById('op2').value);
  var res = document.getElementById('res');
  switch (opt) {
    case '+': res.value = (op1 + op2).toFixed(2);
              break;
    case '-': res.value = (op1 - op2).toFixed(2);
              break;
    case '*': res.value = (op1 * op2).toFixed(2);
              break;
    case '/': res.value = (op1 / op2).toFixed(2);
              break;
  }
}
</script>

<p style='text-align:center'>
<input type='text' size='3' id='op1' value=''>
<input type='button' value='+' onClick="operacao('+')">
<input type='button' value='-' onClick="operacao('-')">
<input type='button' value='*' onClick="operacao('*')">
<input type='button' value='/' onClick="operacao('/')">
<input type='text' size='3' id='op2' value=''> =
<input type='text' size='3' id='res' value=''>
</p>
```

Resultado:



Note que:

- Diferentemente do exemplo anterior, as variáveis `op1` e `op2` já guardam o valor contido na caixa de texto e não o seu endereço. Já a variável `res`, por conveniência, guarda o endereço da caixa onde será escrita a resposta.

- O método `parseFloat()` é utilizá-lo para garantir que os valores digitados nas caixas sejam transformados em números. Se isso não for feito, a soma de '2' com '3' dá '23' e não 5, como desejado.

`input type="range"`

Elementos do tipo `range` correspondem a barras deslizantes normalmente utilizadas para varrer continuamente os possíveis valores de uma variável. O exemplo a seguir mostra como utilizar esse elemento para controlar a posição de uma divisão móvel dentro de uma divisão fixa.

Listagem:

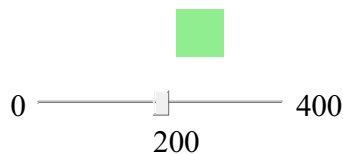
```
<div id="rngDiv" style="width:400;margin:auto"></div>

<div align="center" style="margin-top:1em">
0 <input id="rngCtrl" type="range" min="0" max="10" value="5"
    onChange="atualizaRange()" /> 400
</div>

<div align="center" id="rngValue"></div>

<script>
function atualizaRange() {
    var x = parseInt(document.getElementById("rngCtrl").value)*40;
    document.getElementById("rngValue").innerHTML = x;
    var estilo = "position:relative;";
    estilo += "left:" + x + ";";
    estilo += "width:25px;height:25px;background-color:lightgreen";
    var str = "<div style=" + estilo + "></div>";
    document.getElementById("rngDiv").innerHTML = str;
}
atualizaRange();
</script>
```

Resultado:



Na seção HTML do exemplo são definidas três divisões, todas centralizadas. A primeira divisão, `rngDiv`, de coordenadas fixas no documento, foi criada vazia para receber dentro dela a divisão móvel (sem `id`) a ser construída dentro do script quando a posição da barra for modificada. A segunda divisão (também sem `id`) foi criada apenas para conter e centralizar a barra deslizante e seus valores extremos (0 e 400). A terceira divisão, `rngValue` foi criada para receber dentro dela o valor da posição atual do cursor da barra deslizante.

Note que são utilizadas duas maneiras distintas de centralizar uma divisão. Na primeira divisão isso é feito atribuindo-se à divisão um tamanho e impondo que as margens devem ser automaticamente escolhidas, o que leva à centralização do conteúdo. Na segunda e na terceira divisões isso é feito atribuindo o valor `center` ao parâmetro `align` do elemento `div`.

Na seção que contém o script é definida a função `atualizaRange()`, que responde ao evento `onChange` da barra deslizante. Após a definição, a função é chamada uma vez para que apareça no documento, ao ser carregado, a divisão móvel e sua respectiva posição na barra deslizante.

A função `atualizaRange()` começa atribuindo à variável `x` o valor corrente do controle deslizante, multiplicado por 40. Na instrução seguinte, atribui à `div rngValue` esse valor, para informar ao usuário a posição corrente do cursor. Em seguida, na variável `estilo` é construída uma declaração de estilo que (a) estabelece que as coordenadas que lhe serão atribuídas devem ser interpretadas como relativas à posição da divisão que a contém; (b) ajusta a largura e (c) a altura da divisão a ser inserida dinamicamente no documento para 25 pixels e finalmente (d) dá-lhe um verde claro como cor de fundo. A penúltima linha da função constrói a divisão móvel que será incluída na divisão fixa através da instrução na última linha da função.

Note que os parâmetros `min`, `max` e `value` do `input type="range"` somente admitem valores inteiros, o que determina seus pontos de parada. No caso, como os extremos foram definidos entre 0 e 10, existem 11 paradas (0, 1, 2, ..., 10), que correspondem às posições 0, 40, 80, ..., 400 relativas ao início da divisão fixa.

Sua vez... (9-4)

Modifique o exemplo anterior de modo a atribuir uma altura de 400 pixels para a divisão fixa e incluir uma segunda barra deslizante que controle a posição vertical da divisão móvel.

`input type="checkbox"`

O elemento `checkbox` permite ao usuário escolher uma ou mais alternativas de um grupo, como no exemplo abaixo.

Listagem:

```
<script>
function checkBoxes() {
    var a1 = document.getElementById("cb1").checked;
    var a2 = document.getElementById("cb2").checked;
    var a3 = document.getElementById("cb3").checked;
    var a4 = document.getElementById("cb4").checked;
    if (a1&&a2&&a4&&!a3) alert("Parabéns!");
    else alert("Tente novamente!");
}
</script>

<p>Quais das seguintes unidades são unidades de energia?
<ol>
<input type='checkbox' id='cb1'> joule <br>
<input type='checkbox' id='cb2'> erg <br>
<input type='checkbox' id='cb3'> watt <br>
```

```
<input type='checkbox' id='cb4'> kwh <br>
</ul>
<input type='button' value=' OK ' onClick='checkBoxes()'>
```

Resultado:

Quais das seguintes unidades são unidades de energia?

- a. joule
- b. erg
- c. watt
- d. kwh

OK

Neste exemplo, são criados quatro elementos `input type="checkbox"`, cada um com o seu próprio `id`, através do qual pode ser inequivocamente identificado. Quando o usuário clica no botão OK, a função `checkBoxes()` é acionada. Essa função cria quatro variáveis que recebem os estados de cada um dos elementos. Caso o elemento tenha sido marcado pelo usuário, a chamada a `document.getElementById(id).checked` retorna `true`; caso contrário, retorna `false`. O `if` combina as quatro respostas com operadores "E" (`&&`) e negação (`!`) para verificar se o usuário marcou as respostas corretas e não marcou a resposta errada.

Sua vez... (9-5)

Modifique o exemplo anterior de modo que atribua uma nota de 0 a 3 para o usuário de duas maneiras: (a) sem prejuízo quando o usuário marcar a alternativa incorreta e (b) com prejuízo quando o usuário marcar a alternativa incorreta (isto é, se todas as alternativas forem marcadas a nota é 2).

Uma possível aplicação deste tipo de estrutura é contabilizar o número de escolhas que o usuário faz. No exemplo a seguir, não existem respostas certas ou erradas, apenas deseja-se contar o número de opções escolhidas.

Listagem:

```
<script>
function contaBoxes() {
  var cnt = 0;
  for (var i=1;i<=5;i++) {
    var str = "e" + i;
    var box = document.getElementById(str);
    if (box.checked) cnt++;
  }
  alert(cnt);
}
```



```

</script>

<p>Quais dos seguintes elementos químicos, em estado puro,
você já <b>viu</b> com seus próprios olhos?

<ol type='a'>
<li><input type='checkbox' id='e1'> hidrogênio <br>
<li><input type='checkbox' id='e2'> carbono <br>
<li><input type='checkbox' id='e3'> oxigênio <br>
<li><input type='checkbox' id='e4'> ferro <br>
<li><input type='checkbox' id='e5'> ouro <br>
</ol>

<input type='button' value=' OK ' onClick='contaBoxes()'>

```

Resultado:

Quais dos seguintes elementos químicos, em estado puro, você já viu com seus próprios olhos?

- a. hidrogênio
- b. carbono
- c. oxigênio
- d. ferro
- e. ouro

OK

Sua vez.. (9-6)

Modifique o exemplo anterior de modo a incluir nos elementos `input` o atributo `value`, atribuindo a eles os valores dos números atômicos dos elementos. Utilize o comando `document.getElementById(...).value` para coletar essa informação dos itens marcados e um `window.alert()` para apresentar os números atômicos dos elementos observados.

`input type="radio"`

O elemento `radio` é muito semelhante ao `checkbox`, mas permite fazer com que diversos elementos formem um grupo de modo que quando algum deles é marcado, os outros do mesmo grupo ficam desmarcados. O pertencimento a um grupo é definido pelo uso do atributo `name`, como no exemplo a seguir.

Listagem:

```

<script>
function checkradio() {
    if (document.getElementById("r3").checked)

```

```

        alert("Parabéns!");
    else
        alert("Tente novamente!");
    }
</script>

<p>Qual das seguintes unidades NÃO é uma unidade de energia?
<ol>
<input type='radio' name='grupo' id='r1'> joule <br>
<input type='radio' name='grupo' id='r2'> erg <br>
<input type='radio' name='grupo' id='r3'> watt <br>
<input type='radio' name='grupo' id='r4'> kwh <br>
</ol>
<input type='button' value=' OK ' onClick='checkradio()'>

```

Resultado:

Qual das seguintes unidades NÃO é uma unidade de energia?

- joule
- erg
- watt
- kwh

OK

Este elemento pode ser utilizado para avaliar, por exemplo, o número de questões que o usuário acertou ou algo semelhante. No exemplo abaixo, é simulada uma pergunta para a qual o usuário dá uma nota entre 1 e 3 a diferentes itens de uma pesquisa e obtém a pontuação total ao clicar no botão.

Listagem:

```

<script>
function pontua() {
    var total = 0;
    for (var i=1;i<=3;i++) {
        for (var j=1;j<=3;j++) {
            var str = "q" + i + j;
            var item = document.getElementById(str);
            if (item.checked) total = total + parseInt(item.value);
        }
    }
    alert(total);
}
</script>

<table style='text-align:center; width:400'>
<tr>
<th>Item</th>
<th>Ruim</th>
<th>Regular</th>
<th>Bom</th>
</tr>
<tr>
<td>As instalações físicas são adequadas?</td>

```

```

        <td><input type='radio' name='q1' id='q11' value='1'></td>
        <td><input type='radio' name='q1' id='q12' value='2'></td>
        <td><input type='radio' name='q1' id='q13' value='3'></td>
    </tr>
    <tr>
        <td>O livro didático é de boa qualidade?
        <td><input type='radio' name='q2' id='q21' value='1'></td>
        <td><input type='radio' name='q2' id='q22' value='2'></td>
        <td><input type='radio' name='q2' id='q23' value='3'></td>
    </tr>
    <tr>
        <td>O professor está bem preparado?
        <td><input type='radio' name='q3' id='q31' value='1'></td>
        <td><input type='radio' name='q3' id='q32' value='2'></td>
        <td><input type='radio' name='q3' id='q33' value='3'></td>
    </tr>
</table>

<p style='text-align:center'>
<input type='button' value=' OK ' onClick='pontua()'>
</p>

```

Resultado:

Item	Ruim	Regular	Bom
As instalações físicas são adequadas?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
O livro didático é de boa qualidade?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
O professor está bem preparado?	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Cada questão tem seu grupo de respostas formado pela atribuição do mesmo `name` a todos os seus itens e que os `id` são todos diferentes, uma vez que dois elementos do documento não podem ter o mesmo `id`. Na função `pontua()`, foram utilizados os índices `i` e `j` dos dois laços para a construção de um literal `str` que contém cada um desses `id`'s à medida que os laços progredem. Como duas propriedades dos elementos são utilizadas (`checked` e `value`), é conveniente armazenar a referência obtida pela chamada a `getElementById()` em uma variável `item`, que a seguir é utilizadas para a obtenção dos valores das duas propriedades.

A propriedade `value` é sempre interpretada como um literal e é convertida para um número inteiro utilizando `parseInt()`. Caso isso não fosse feito, o resultado de `'2'+3+'1'`, por exemplo, seria `'231'` e não 6, como esperado.

Sua vez... (9-7)

(a) Invente e inclua mais duas perguntas no exemplo anterior e altere a função `pontua()` de modo que o novo script funcione apropriadamente. (b) Modifique os valores dos itens para `-1`, `0` e `1` e adicione instruções à função `pontua()` de modo que emita um aviso (`window.alert()`) com a mensagem

"reprovado" caso a pontuação final seja negativa, "aprovado" caso seja positiva e "em análise" caso seja nula.

input type="select"

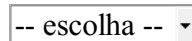
Outro elemento interativo é o `select`, que oferece um menu de opções a serem escolhidas. O exemplo a seguir mostra como estruturar as opções e mostrar a escolha do usuário.

Listagem:

```
<script>
function confirma() {
    var opt = document.getElementById("parts").value;
    if (opt=="eletron") window.alert("Elétron");
    else if (opt=="proton") window.alert("Próton");
}
</script>

<p style='text-align:center'>
<select id="parts" onChange="confirma()">
    <option value="escolha">-- escolha --</option>
    <option value="eletron">Elétron</option>
    <option value="proton">Próton</option>
</select>
</p>
```

Resultado:



Note o uso de um `else if` para evitar que o programa evoque um `window.alert()` quando, ao fazer uma nova escolha, o usuário seleccione `-- escolha --`.

Sua vez.. (9-8)

Modifique o exemplo acima de modo a incluir mais partículas (por exemplo, o nêutron) e fazer com que, quando o usuário voltar a escolher `-- escolha --`, o programa, ao invés de não fazer nada, emita um aviso do tipo "Você deve escolher uma partícula!"

O exemplo a seguir ilustra o uso do `select/option` com outro importante recurso do HTML/JavaScript, a propriedade `innerHTML` do objeto `document`. Esta propriedade pode ser utilizada para modificar o conteúdo dos elementos de um documento depois de ser carregado.

Listagem:

```
<script>
function processa() {
  var opt = document.getElementById("particulas").value;
  var str;
  switch (opt) {
    case "eletron":
      str = "<i>m</i> = 9.11 × 10<sup>-31</sup> kg<br>";
      str += "<i>q</i> = -1.60 × 10<sup>-19</sup> C";
      break;
    case "proton":
      str = "<i>m</i> = 1.67 × 10<sup>-27</sup> kg<br>";
      str += "<i>q</i> = 1.60 × 10<sup>-19</sup> C";
      break;
  }
  document.getElementById("propriedades").innerHTML = str;
}
</script>

<table width="300">
<tr>

  <td style='vertical-align:top; width:150px; height:50px;'>
  <select id="particulas" onChange="processa()">
    <option value="escolha">-- escolha --</option>
    <option value="eletron">Elétron</option>
    <option value="proton">Próton</option>
  </select>
  </td>

  <td style='vertical-align:top; width:150px; height:50px;'>
  <div id="propriedades"></div>
  </td>

</tr>
</table>
```

Resultado:



O exemplo contém um bloco que define a função `processa()` e outro bloco que define uma tabela. A tabela contém apenas uma linha (delimitada por `<tr>...</tr>`) com duas células (delimitadas por `<td>...</td>`). A caixa de escolha `select/option` está dentro da primeira célula. A outra célula contém uma divisão (`<div>...</div>`) vazia, mas identificada (`id="propriedades"`). Esta divisão "reserva" um espaço que permite mostrar conteúdos gerados *depois* que o documento tenha sido carregado pelo navegador. A inserção de conteúdo nesta divisão é feita atribuindo o conteúdo desejado à propriedade `innerHTML` de objeto do documento.

A caixa de seleção tem três opções, sendo a primeira apenas um valor neutro que define o tamanho da caixa (é maior que todas as outras opções) e força o usuário a de fato fazer uma escolha. A função `processa()` usa o método `getElementById()` para obter que escolha fez o usuário, guardando-a na variável `opt`. Um `switch...case` constrói um literal

(`str`) cujo conteúdo depende da escolha do usuário. Finalmente, este conteúdo é atribuído à propriedade `innerHTML`, que coloca o texto no objeto especificado em `getElementById()`, no caso a divisão "propriedades".

Note a diferença entre as duas primeiras linhas dentro de cada `case`: a primeira tem somente um sinal de igual (=), que atribui a literal entre aspas à variável `str`; a segunda tem um sinal de mais seguido por um sinal de igual (+=), que atribui à variável `str` o seu conteúdo atual (definido na linha anterior) concatenado à literal entre aspas (é a forma contraída de `str = str + "..."`).

Sua vez.. (9-9)

Se, no exemplo anterior, você voltar a selecionar -- `escolha` -- verá que na área destinada às informações sobre a partícula aparecerá o termo `undefined`. Modifique a função `processa()` de modo a introduzir mais um `case` que ao invés disso mostre algo como "escolha uma partícula".

textarea

O elemento `textarea` é um recurso para o processamento de textos não formatados com múltiplas linhas. O exemplo a seguir define duas caixas de texto para que o usuário digite o número de linhas e de colunas de uma tabela, uma área de texto para que ele entre com os conteúdos das células, separados por vírgulas e quebras de linha, e uma área de texto onde o script imprime a tabela resultante que pode ser recortada e colada em arquivo HTML.

Listagem:

```
<p>
Nro. de linhas: <input type='text' id='nlin' value='3' size='3'>
Nro. de colunas: <input type='text' id='ncol' value='3' size='3'>
</p>
<p>
<textarea rows='5' id='inarea' cols='60'>
11, 12, 13
21, 22, 23
31, 32, 33
</textarea>
</p>
<p><input type='button' value='FORMATA' onClick='formataTabela()'>
<p>
<textarea rows='7' id='outarea' cols='60'>
</textarea>
</p>

<script>
function formataTabela() {
    var nlin = parseInt(document.getElementById("nlin").value);
    var ncol = parseInt(document.getElementById("ncol").value);
    var intxt = document.getElementById("inarea").value;
    var linhas = (intxt.split("\n")).toString();
    var dados = linhas.split(",");

    var str = "<table>\n";
    for (var i=0;i<nlin;i++) {
```

```

        str += "<tr>";
        for (var j=0;j<ncol;j++) {
            str += "<td>" + dados[i*ncol+j] + "</td>";
        }
        str += "</tr>\n";
    }
    str += "</table>";

    document.getElementById("outarea").value = str;
}
</script>

```

Resultado:

Nro. de linhas: Nro. de colunas:

11, 12, 13
21, 22, 23
31, 32, 33

FORMATATA

```

<table>
<tr><td>11</td><td> 12</td><td> 13</td></tr>
<tr><td>21</td><td> 22</td><td> 23</td></tr>
<tr><td>31</td><td> 32</td><td> 33</td></tr>
</table>

```

Os elementos `textarea` têm dois atributos fundamentais que são o número de linhas (`rows`) e o número de colunas (`col`). No exemplo, eles também têm identificadores únicos (`id`) que são utilizados pelo método `document.getElementById()` para ler ou escrever seu conteúdo.

Depois de buscar nos elementos o número de linhas, de colunas e o texto digitado na caixa de entrada, o script usa o método `split("\n")` do objeto `String` para remover as quebras de linha ("`\n`"). O método retorna uma matriz com três elementos (as três linhas do texto digitado). Na mesma instrução é aplicado a esta matriz o método `toString()`, que transforma a matriz em uma única literal, colocando vírgulas nas junções. O resultado disso é que a variável `linhas` fica com um literal formado pelos conteúdos das células, separados por vírgula. Finalmente é utilizado o método `split(",")` na variável `linhas`, separando seu conteúdo em uma matriz com 9 elementos denominada `dados`. O segundo bloco da função declara a variável `str` que irá receber a tabela formatada.

Note que, como a matriz `dados` é unidimensional, sua indexação é feita utilizando `[i*ncol+j]`, que tem o mesmo efeito que `[i][j]` teria no caso de uma matriz bidimensional.

Sua vez.. (9-10)

Experimente utilizar o exemplo anterior colocando na área de entrada (a) todos os dados em uma única linha, separados por vírgula e (b) um dado em cada linha. O algoritmo funciona corretamente? Que modificações você faria para que funcionasse **exclusivamente** em um caso ou no outro?

map/usemap

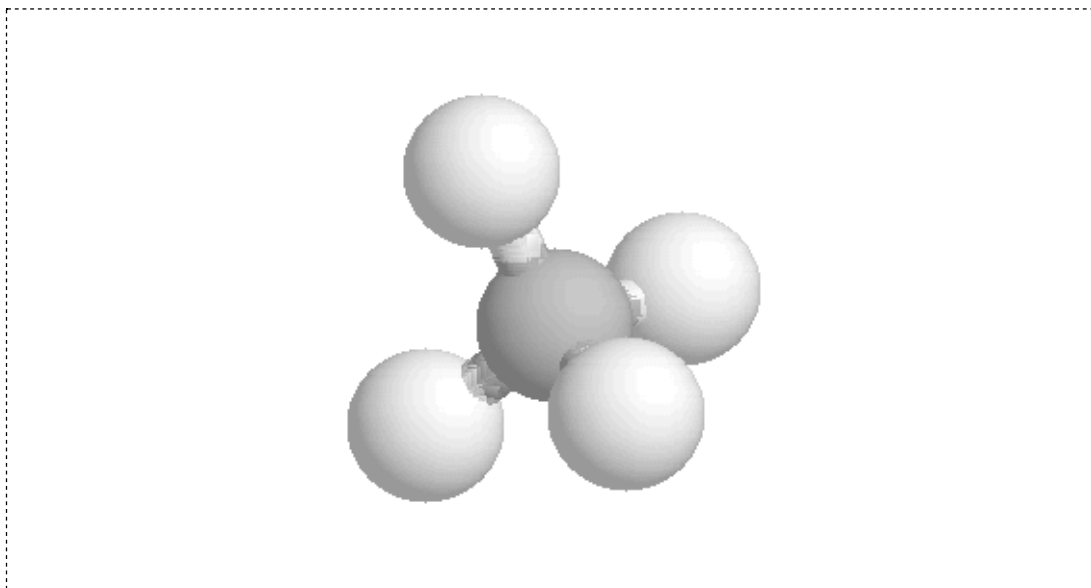
Mapas são uma maneira de dar interatividade a uma figura. O exemplo abaixo mostra uma figura que contém uma representação de uma molécula de metano (CH₄). Quando o usuário clica sobre um dos "átomos", uma janela de alerta (`window.alert()`) é apresentada com o nome do átomo:

Listagem:

```
<p style='text-align:center'>
<map name="mapa">
<area shape="circle" coords="130,140,20"
      onClick="window.alert('Carbono')" >
<area shape="circle" coords="91,60,20"
      onClick="window.alert('Hidrogênio')" >
<area shape="circle" coords="200,120,20"
      onClick="window.alert('Hidrogênio')" >
<area shape="circle" coords="170,190,20"
      onClick="window.alert('Hidrogênio')" >
<area shape="circle" coords="60,190,20"
      onClick="window.alert('Hidrogênio')" >
</map>

</p>
```

Resultado:



O atributo `area` pode receber quatro valores: `default` (toda a figura), `rect`

(retângulo), `circle` (círculo) e `poly` (polígono). Para um retângulo, o atributo `coords` deve receber `x1, y1, x2, y2` que são as coordenadas, em pixels, do canto superior esquerdo e do canto inferior direito da região mapeada; para um círculo, deve receber `xc, yc, r`, as coordenadas do centro do círculo e seu raio; para um polígono, deve receber `x1, y1, x2, y2, ..., xn, yn`, as coordenadas dos vértices do polígono. O mapa deve ter um nome (`name`) que será utilizado pelo `usemap` (com o caracter "#" na frente) no elemento que especifica a figura.

As coordenadas das regiões de interesse da figura podem ser obtidas utilizando-se um editor de imagens tipo Paint, Gimp e similares.

Sua vez... (9-11)

Substitua a imagem do exemplo por uma foto de seu gosto e acrescente um mapa de elementos identificando pessoas, objetos, lugares etc.

`clientX`, `clientY`

A posição do mouse é um outro elemento de interação que pode ser útil. O script a seguir mostra como fazer para obter a posição do mouse no documento e mostrá-la em duas caixas de texto:

Listagem:

```
<script>
document.onmousemove = getMouseXY;

function getMouseXY(event) {
    document.getElementById("MouseX").value = event.clientX;
    document.getElementById("MouseY").value = event.clientY;
}
</script>

<p style="text-align:center">
<i>x</i>: <input type="text" id="MouseX" value="0" size="3">
<i>y</i>: <input type="text" id="MouseY" value="0" size="3">
</p>
```

Resultado:

x: y:

O script começa adicionando ao método padrão `onmousemove` a função `getMouseXY`, definida pelo usuário. A função definida pelo usuário recebe como parâmetro uma referência ao objeto `event`, que traz informações sobre eventos detetados pelo navegador. O objeto `event` tem algumas propriedades, entre elas a posição `x,y` do mouse, armazenadas em

clientX e clientY.

As coordenadas fornecidas são relativas ao canto superior esquerdo da parte do documento que aparece na tela. Isto significa que se você mover o documento utilizando a barra de rolagem o referencial das coordenadas muda.

Sua vez.. (9-12)

Modifique o exemplo anterior de modo que a função emita um aviso quando o mouse (a) estiver em uma região compreendendo aproximadamente o terço central da tela e (b) estiver fora dessa região (se sua tela tiver 800 pixels de largura e 600 de altura, o terço central será **aproximadamente** definido pelo retângulo cujos cantos superior esquerdo e inferior direito têm coordenadas (270,200) e (540,400), respectivamente.

No exemplo a seguir, os métodos `onmousedown` (pressionar um botão do mouse) e `onmouseup` (soltar um botão do mouse) são substituídos por funções especificadas pelo usuário. A função `onmousedown` guarda as coordenadas do ponto onde o botão do mouse foi abaixado e a função `onmouseup` guarda as coordenadas do ponto onde o botão foi liberado, calculando e apresentando a distância, em pixels, entre esses pontos.

Listagem:

```
<script>
document.onmousedown = mouseDown;
document.onmouseup = mouseUp;

var iniX, iniY, fimX, fimY;

function mouseDown(event) {
    iniX = event.clientX;
    iniY = event.clientY;
}

function mouseUp(event) {
    fimX = event.clientX;
    fimY = event.clientY;
    var dist = (fimX-iniX)*(fimX-iniX) + (fimY-iniY)*(fimY-iniY);
    dist = Math.sqrt(dist);
    window.alert(dist.toFixed(0) + " pixels");
}
</script>
```

Sua vez.. (9-13)

Elimine do exemplo anterior as funções `mouseUp(event)` e `mouseDown(event)` substituindo-as por uma única função `mouseClick(event)` que deve ser atribuída ao gerenciador padrão de cliques do documento através da

instrução `document.onclick = mouseClick;`. Essa função deve capturar as coordenadas de dois cliques distintos e mostrar a distância entre eles. *Dica:* crie uma variável global booleana cujo valor (`true` ou `false`) é verificado a cada clique: se for `true`, armazena as coordenadas do primeiro ponto e armazena a variável para `false`; se for `false`, armazena as coordenadas do segundo ponto e ajusta a variável para `true`.

O exemplo a seguir ilustra como transformar as coordenadas do mouse, fornecidas pelo navegador, para o sistema de coordenadas de uma figura arbitrária.

Listagem:

```
<p>
<table align="center" cellpadding="10">
<tr>
<td>
<p style="text-align:center; border:1px solid;">

</p>
</td>
<td>
<p>
Canto superior esquerdo:<br>
<i>x</i>: <input type="text" id="xCH41" value="0" size="3">
<i>y</i>: <input type="text" id="yCH41" value="0" size="3">
</p>
<p>
Canto inferior direito:<br>
<i>x</i>: <input type="text" id="xCH42" value="0" size="3">
<i>y</i>: <input type="text" id="yCH42" value="0" size="3">
</p>
<p style="text-align:center">
<input type="button" value="CALIBRAR" onClick="calibraCH4()">
</p>
<p>
Posição do mouse descalibrada:<br>
<i>x</i>: <input type="text" id="xCH4" value="0" size="3">
<i>y</i>: <input type="text" id="yCH4" value="0" size="3">
</p>
<p>
Posição do mouse calibrada:<br>
<i>x</i>: <input type="text" id="xCH4Cal" value="0" size="3">
<i>y</i>: <input type="text" id="yCH4Cal" value="0" size="3">
</p>
</td>
</tr>
</table>
</p>

<script>

var AxCH4 = 0;
var BxCH4 = 1;
var AyCH4 = 0;
var ByCH4 = 1;
document.onmousemove = getMouseCH4;

function getMouseCH4(event) {
    var docX = event.clientX;
    var docY = event.clientY;
```

```

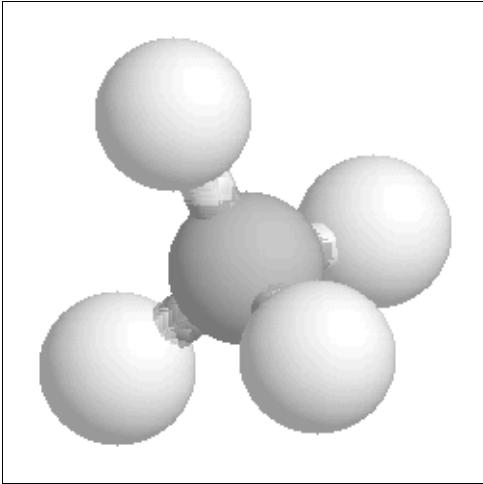
document.getElementById("xCH4").value = docX
document.getElementById("yCH4").value = docY;
var calX = AxCH4 + BxCH4 * docX;
var calY = AyCH4 + ByCH4 * docY;
document.getElementById("xCH4Cal").value = calX.toFixed(0);
document.getElementById("yCH4Cal").value = calY.toFixed(0);
}

function calibraCH4() {
  var xMse1 = parseFloat(document.getElementById("xCH41").value);
  var yMse1 = parseFloat(document.getElementById("yCH41").value);
  var xMse2 = parseFloat(document.getElementById("xCH42").value);
  var yMse2 = parseFloat(document.getElementById("yCH42").value);
  var xFig1 = 0;
  var yFig1 = 0;
  var xFig2 = 250;
  var yFig2 = 250;
  BxCH4 = (xFig2-xFig1)/(xMse2-xMse1)
  AxCH4 = xFig1 - BxCH4 * xMse1;
  ByCH4 = (yFig2-yFig1)/(yMse1-yMse2);
  AyCH4 = yFig1 - ByCH4 * yMse2;
}

</script>

```

Resultado:



Canto superior esquerdo:
x: y:

Canto inferior direito:
x: y:

Posição do mouse descalibrada:
x: y:

Posição do mouse calibrada:
x: y:

Na primeira parte da listagem está o conteúdo HTML que define as caixas de texto utilizadas para introduzir e apresentar informações; na segunda parte está o script de calibração propriamente dito. O usuário deve posicionar o mouse nos cantos superior esquerdo e inferior direito da figura, obter as coordenadas com relação ao documento, digitá-las nos quadros correspondentes e pressionar o botão CALIBRAR. Feito isto, serão apresentadas tanto as coordenadas com relação ao canto superior esquerdo da parte visível do documento (a janela "cliente") quanto as coordenadas com relação ao canto inferior esquerdo da figura, em pixels.

Sua vez.. (9-14)

Modifique o exemplo anterior de modo que a calibração seja feita relativamente ao centro da molécula.

keyCode

É possível obter o código de uma tecla pressionada e realizar alguma ação em função do seu valor. O exemplo a seguir mostra o uso de `onkeydown` no marcador `<body>` do HTML para chamar a função `passoTecla(event)` toda vez que uma tecla estiver abaixada. Quando as telas "seta para a esquerda" (37) e "seta para a direita" (39) forem pressionadas, uma seta (← ou →) vai andar para a esquerda ou para a direita dentro de uma divisão; se qualquer outra tecla for pressionada, um × é mostrado parado na última posição.

Listagem:

```
<html>

<body onkeydown="mostraTecla(event)">

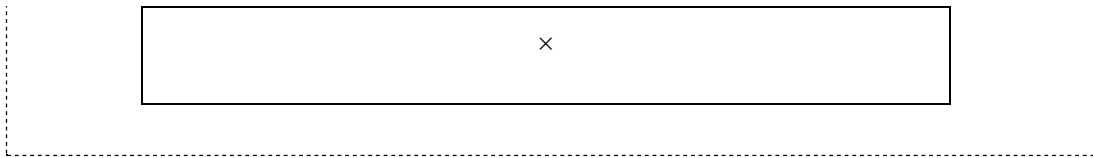
<script>
var passoTecla = 0;

function mostraTecla(evnt)
{
    if (window.event) tecla = evnt.keyCode;    // Internet Explorer
    else tecla = evnt.which;                  // Firefox, Chrome
    var txt = "&times;";
    if (tecla==37) {
        passoTecla += -10;
        txt = "&larr;";
    }
    if (tecla==39) {
        passoTecla += 10;
        txt = "&rarr;";
    }
    if (passoTecla>200) passoTecla = -200;
    if (passoTecla<-200) passoTecla = 200;
    var str = "<div style='position:relative; left:" + passoTecla + ";'>";
    str += txt;
    str += "</div>";
    document.getElementById("teclas").innerHTML = str;
}
</script>

<p>
<div id="teclas" style="border: 1px solid black; margin-left:60px;
width: 420px; height: 40px; text-align:center; padding-top:10px;">
</div>
</p>
</div>

</body>
</html>
```

Resultado:



Neste exemplo foi novamente utilizado o atributo `innerHTML` para inserir conteúdo em uma divisão do documento. Neste caso, foi inserida uma nova divisão com os atributos de estilo `position` e `left`, sendo o valor desse último controlado ao pressionar-se a tecla de seta para a direita ou para a esquerda.

Note que é feita uma verificação da coordenada da divisão, fazendo com que a seta "circule" pela caixa caso o valor do atributo de estilo `left`, armazenado na variável `passoTecla`, atinja valores tais que a divisão seria posicionada além dos limites da caixa.

Sua vez.. (9-15)

Modifique o exemplo anterior de modo a ampliar a altura da caixa e fazer com que o elemento flutuante responda também às teclas "seta para cima" e seta para baixo" (cujos códigos podem ser obtidos facilmente na internet).

Exercícios

1. Modifique o script da calculadora de modo a incluir botões para as seguintes funções: seno, cosseno, tangente, exponencial, logaritmo natural, logaritmo na base 10, raiz quadrada, raiz cúbica e função potência.
2. Refaça a sua calculadora utilizando somente as funções que recebem apenas um argumento. Ao invés de botões, utilize um grupo de controles `radio` para a escolha da função desejada. Utilize o `onChange` em cada item da sua lista de funções para que, quando a escolha for feita, o valor da função seja apresentado. Seu script deve gerar algo parecido com:

$x =$ $\text{sen}(x)$ $\text{cos}(x)$ $\text{tan}(x)$ $f(x) =$

3. Faça um script que contenha quatro campos de texto e um botão. O usuário deve escrever ou colar em um dos campos de texto uma sequência de números inteiros arbitrários e, quando o usuário pressionar o botão, seu script deve escrever nos outros três campos o maior número, o menor número e a média dos números. *Dica 1:* Estude o método `split()` e utilize-o para transformar o texto digitado em uma matriz unidimensional de números inteiros. *Dica 2:* O script a seguir mostra uma possível estratégia para obter o maior número em uma matriz unidimensional.

```
<script>
var mat = new Array(5,3,2,7,4); // matriz 1D com os números
var maxNro = -Infinity;       // o menor número possível
```

```

for (var i=0; i<mat.length; i++) {
    // se o nro for maior que o max atual é o novo max
    if (maxNro < mat[i]) maxNro = mat[i];
}
window.alert(maxNro);
</script>

```

4. Elabore uma prova com cinco questões de múltipla escolha, utilizando elementos tipo `radio`. O usuário deve responder às questões e, ao clicar em um botão "OK", receber uma nota entre 0 e 10 proporcional ao número de acertos.
5. Elabore uma questão do tipo "marque verdadeiro ou falso" com 5 alternativas, utilizando elementos do tipo `checkbox`. O script deve imprimir a nota final obtida a partir das escolhas do usuário de modo que, para cada acerto (verdadeiro ou falso), o usuário ganhe 2 pontos e, para cada erro, perca dois pontos, sendo que a nota final não pode ser negativa.
6. A *impedância acústica* Z de um material é o produto da densidade ρ , medida em kg/m^3 , com a velocidade do som v no meio, medida em m/s : $Z = \rho \times v$ (em $\text{kg/m}^2/\text{s}$). O *coeficiente de reflexão* de uma interface, para uma incidência normal (90 graus) do ultrassom, é:

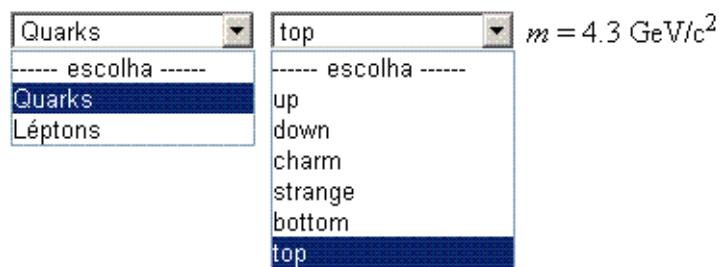
$$R = \left(\frac{Z_1 - Z_2}{Z_1 + Z_2} \right)^2$$

onde Z_1 e Z_2 são as impedâncias acústicas dos meios que definem a interface. Faça um script que apresente uma tabela em que o usuário possa escolher os tecidos que definem uma interface e calcular a fração de ultrassom refletida nela:

Material 1	Material 2	Impedância (kg/m ² /s × 10 ⁶)
<input checked="" type="radio"/> Ar	<input type="radio"/> Ar	0,0004
<input type="radio"/> Gordura	<input checked="" type="radio"/> Gordura	1,38
<input type="radio"/> Água	<input type="radio"/> Água	1,54
<input type="radio"/> Músculos	<input type="radio"/> Músculos	1,70
<input type="radio"/> Ossos	<input type="radio"/> Ossos	7,8

Calcula reflexão R = %

7. Modifique o exemplo do `select/option` com elétrons e prótons de modo que as opções passem a ser "quarks" e "léptons". Ao escolher um tipo, deve aparecer uma nova caixa de escolhas com as opções "up", "down", "charm", "strange", "bottom" e "top" para os quarks ou com as opções "elétron", "múon", "tau", "eletron-neutrino", "muon-neutrino" e "tau-neutrino" para os léptons. Quando o usuário escolher um item desta segunda caixa, deve aparecer em um terceiro campo a massa das partículas (retorne ao capítulo sobre tabelas para obter os valores das massas dos quarks e dos léptons). Seu script deve gerar algo parecido com:



8. Uma maneira de fazermos um gráfico enquanto não aprendemos como lidar com elementos de desenho em HTML/JavaScript é fazê-lo com caracteres. Faça um script que contenha uma área de texto para entrada de dados, um botão e uma área de texto para saída de dados. Na área de dados, o usuário digita os valores da coordenada y do gráfico, separados por vírgula. Ao clicar no botão, o script deve imprimir na primeira linha da área de saída de dados um número de asteriscos igual ao valor do primeiro número digitado, na segunda linha um número de asteriscos igual ao valor do segundo número digitado e assim por diante. Seu script deve ter uma aparência aproximadamente igual a esta:

1,2,4,9,16,9,4,2,1

GRÁFICO


```
*
**
***
****
*****
*****
*****
****
***
**
*
```

9. Modifique o exemplo sobre a calibração da posição do mouse sobre a figura de modo que os pontos de referência (canto superior esquerdo e canto inferior direito) não precisem ser digitados, sendo obtidos com cliques do mouse.