

5

As janelas do browser

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, transferir informações entre janelas e frames, abrir e fechar novas janelas e criar janelas de diálogo.

Janelas do browser são representadas em JavaScript através de objetos do tipo *Window*. Pode-se classificar as janelas usadas em JavaScript em cinco categorias:

- *Janela da aplicação*: é um papel assumido pela última janela aberta do browser. Se esta janela for fechada, a aplicação é encerrada. Em JavaScript, métodos para fechar janelas (`close()`) não funcionam na última janela.
- *Janelas abertas através de instruções JavaScript*: são novas janelas abertas através de um método `open()`. Podem ter tamanho e características diferentes, ser manipuladas e manipular a janela que as criou, recebendo ou retornando dados, lendo ou alterando propriedades, invocando métodos, inclusive para fechar a outra janela.
- *Janelas abertas através de HTML*: são janelas abertas usando links com o descritor `target` (``). JavaScript pode carregar novas páginas nessas janelas, mas não pode manipular suas propriedades ou métodos.
- *Janelas estruturais*: são janelas ou *frames* que contém uma página HTML que estabelece uma estrutura que divide a janela em *frames* (contém bloco `<FRAMESET>` e não contém `<BODY>`). Possui referências para cada frame que contém.
- *Frames de informação*: são *frames* de uma janela pai que contém uma página HTML com informação (contém um bloco `<BODY>`). Este tipo de janela só possui referências para as janelas que as contém.

Além das janelas comuns, que contém páginas HTML, há três janelas de diálogo: *alerta*, *confirmação* e *entrada de dados*, que não têm propriedades manipuláveis. Todos os tipos de janelas são representadas através de propriedades do objeto `window`.

Objeto Window

O tipo *Window*¹ representa janelas. A propriedade global `window` representa a *janela do browser onde roda o script*. Através de `window`, têm-se acesso a outras propriedades que referenciam possíveis sub-janelas, a janela que a criou (se existir) ou *frames*. Também têm-se acesso a métodos que abrem caixas de diálogo de aviso, confirmação e entrada de dados.

As propriedades e métodos de *Window*, quando referentes à janela atual (objeto `window`), podem omitir o nome do objeto:

```
window.status = "oye!";           // ou status = "oye!";
window.open("documento.html");    // ou open("documento.html");
```

Mas isto só vale se a janela na qual se deseja invocar o método ou a propriedade for a janela atual, onde roda o script. A propriedade `window` refere-se sempre à janela atual.

A tabela abaixo relaciona as propriedades dos objetos do tipo *Window*. Observe que muitos são objetos *Window* e, como consequência, têm as mesmas propriedades:

Propriedade	Acesso	Função
<code>defaultStatus</code>	read / write	Contém <i>String</i> Texto que aparece por <i>default</i> na barra de status da janela.
<code>status</code>	r / w	Contém <i>String</i> Define texto que aparecerá na barra de status.
<code>name</code>	r / w	Contém <i>String</i> Contém nome da janela. Este nome é utilizável em HTML no atributo <code>TARGET</code> em <code></code> e em <code><BASE TARGET="nome"></code> . Em <i>frames</i> , retorna uma referência <i>Window</i> .
<code>document</code>	r	Contém <i>Document</i> . Referência à página contida na janela.
<code>history</code>	r	Contém <i>History</i> . Referência ao histórico da janela.
<code>location</code>	r	Contém <i>Location</i> . Referência à URL exibida na janela.
<code>navigator</code>	r	Contém <i>Navigator</i> . Referência a string de identificação do <i>browser</i> .
<code>opener</code>	r	Contém <i>Window</i> . Refere-se a janela que abriu esta janela
<code>self</code>	r	Contém <i>Window</i> . Referência à própria janela. Mesmo que <code>window</code>
<code>window</code>	r	Contém <i>Window</i> . Sinônimo de <code>self</code> .
<code>frames</code>	r	Contém <i>Array</i> de <i>Window</i> . Vetor dos frames contidos na janela.
<code>length</code>	r	Contém <i>Number</i> . Número de elementos <i>Window</i> no vetor <code>frames</code> (mesma coisa que <code>window.frames.length</code>)
<code>parent</code>	r	Contém <i>Window</i> . Referência à janela que contém esta janela (só existe quando a janela atual é um frame)
<code>top</code>	r	Contém <i>Window</i> . Referência à janela <i>que não é frame</i> que contém a janela atual (só existe quando a janela atual é um <i>frame</i>)


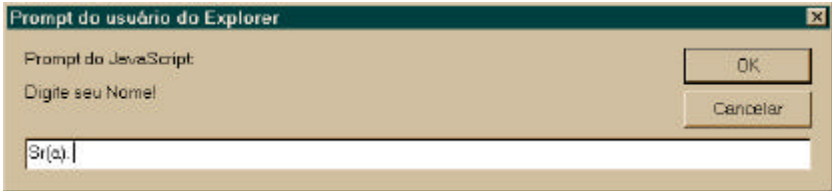
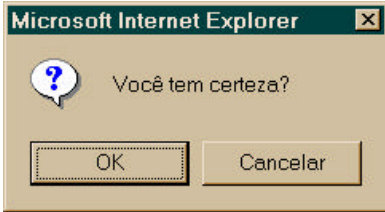
¹ 'Window' é um nome genérico que usamos para qualificar janelas. Não há construtor ou qualquer propriedade com este nome. Existe sim, a propriedade `window` (com "w" minúsculo), que representa a janela atual.

As propriedades `top`, `frames`, `length` e `parent` só têm sentido em janelas que são frames ou que estão dentro de frames. A propriedade `opener` só existe em janelas que foram abertas por outras janelas. É uma forma da 'janela filha' ter acesso à sua 'janela mãe'.

Além das propriedades acima, *Window* possui vários métodos com finalidades bem diferentes. Com eles é possível criar de janelas de diálogo e janelas do browser com aparência personalizada, manipular janelas e realizar tarefas pouco relacionadas com janelas como rolamento de páginas e temporização.

Janelas de diálogo

Três métodos de *Window* são usados apenas para criar janelas de diálogo. Eles são: `alert()`, `confirm()` e `prompt()` e estão listados na tabela abaixo. Não é possível retornar ao controle da janela (de onde foram chamados) sem que os diálogos sejam fechados.

Método	Exemplo
<code>alert("msg")</code>	<pre>window.alert("Tenha Cuidado!");</pre> 
<code>prompt("msg")</code> ou <code>prompt("msg", "texto inicial")</code>	<pre>nome = window.prompt("Digite seu Nome!", "Sr(a). ");</pre>  <p>Retorna <i>String</i>. Devolve o string digitado caso o usuário clique em OK e um string nulo caso o usuário clique em Cancelar.</p>
<code>confirm("msg")</code>	<pre>if (window.confirm("Você tem certeza?")) { ... }</pre>  <p>Retorna <i>Boolean</i>: true caso o usuário clique em OK e false caso o usuário clique em Cancelar.</p>

Nos exemplos acima, a referência `window` pode ser omitida ou substituída por outra referência caso os diálogos estejam sendo abertos em outras janelas.

Métodos para manipular janelas

Os métodos restantes definidos para os objetos *Window*, manipulam parâmetros das janelas, abrem e fecham novas janelas, rolam a página e definem funções de temporização. Estão listados na tabela abaixo.

Método	Ação
<code>open("URL")</code> ou <code>open("URL", "nome")</code> ou <code>open("URL", "nome", "características")</code>	<p>Abre uma nova janela contendo um documento indicado pela URL. Opcionalmente, a janela pode ter um <i>nome</i> que pode ser usado em HTML, ou ter alteradas características como tamanho, layout, etc. (veja tabela abaixo). Retorna uma referência do tipo <i>Window</i> para a janela criada:</p> <pre>filha = window.open("http://a.com/abc.html");</pre>
<code>close()</code>	Fecha uma janela (não vale para frames para a última janela da aplicação).
<code>blur()</code>	Torna uma janela inativa
<code>focus()</code>	Torna uma janela ativa (traz para a frente das outras, se for uma janela independente).
<code>scroll(x, y)</code>	Rola o documento dentro de uma janela de forma que as coordenadas <i>x</i> e <i>y</i> (em pixels) da página apareçam no canto superior esquerdo da área útil da janela, se possível.
<code>setTimeout("instruções", atraso)</code>	<p>Executa uma ou mais instruções JavaScript após um período de atraso em milissegundos. Este método é parecido com a função <code>eval()</code>, mas com temporização. O código continua a ser interpretado imediatamente após o <code>setTimeout()</code>. A espera ocorre em um <i>thread</i> paralelo. Retorna <i>Number</i>: um número de identificação que pode ser passado como argumento do método <code>clearTimeout()</code> para executar a operação imediatamente, ignorando o tempo que falta.</p>
<code>clearTimeout(id)</code>	Cancela a temporização de uma operação <code>setTimeout()</code> cujo número de identificação foi passado como parâmetro, e faz com que as instruções do <code>setTimeout()</code> sejam interpretadas e executadas imediatamente.

Uma janela pode ser aberta em qualquer lugar. Basta fazer:

```
window.open("documento.html"); // ou simplesmente open("documento.html");
```

Janelas com aparência personalizada

As janelas abertas podem ter várias de suas características alteradas no momento em que são abertas. Estas características deverão vir em uma string com uma lista de opções separadas por vírgulas, como o terceiro argumento opcional do método `open()`. Cada característica pode ou não ter um valor. *Não deverá haver espaços* em qualquer lugar da lista. Por exemplo:

```
window.open("enter.html", "j2", "height=200,width=400,status");
```

abre uma janela de 200 pixels de altura por 400 de largura *sem* barra de ferramentas, *sem* barra de diretórios, *sem* campo de entrada de URLs, *sem* barra de menus, não-redimensionável e com barra de status. As características estão na tabela abaixo:

Característica	Resultado
height=h	h é a altura da janela em pixels: height=150
width=w	w é a largura da janela em pixels: width=300
resizable	Se estiver presente permite redimensionar a janela
toolbar	Se estiver presente, mostra a barra de ferramentas do browser
directories	Se estiver presente, mostra a barra de diretórios do browser
menubar	Se estiver presente, mostra a barra de menus do browser
location	Se estiver presente, mostra o campo para entrada de URLs
status	Se estiver presente, mostra a barra de status

Se for utilizado o método `open()` com três argumentos, qualquer característica acima que não apareça listada no string passado como terceiro argumento, não estará presente.

Propriedades da barra de status

A propriedade `defaultStatus` determina o valor *default* do texto que é exibido na barra de status do browser. Geralmente este valor é um string vazio (" ") mas pode ser alterado. A propriedade `status` é usada para mudar o valor da barra de status no momento em que um novo valor é atribuído. Para fazer links informativos, que apresentam uma mensagem na barra de status quando o mouse passa sobre eles, pode-se usar:

```
<script>
    window.defaultStatus=" ";
</script>

<a href="resultados.html" onmouseover="window.status='Resultados' "
    onmouseout="window.status = window.defaultStatus">
    Clique Aqui!</a>
```

Uma aplicação comum para o `window.status` é colocar uma mensagem rolando na barra de status. O processo é semelhante ao proposto para um campo de textos, no exercício 4.1. Consiste em colocar o primeiro caractere no final de uma string e escrevê-lo no `window.status`.

Eventos

Vários eventos do JavaScript estão relacionados com janelas. Estes eventos são chamados a partir dos atributos HTML listados abaixo, que são aplicáveis aos descritores HTML `<BODY>` e `<FRAME>`:

- `ONBLUR` – quando a janela deixa de ser a janela ativa
- `ONERROR` – quando ocorre um erro (uma janela deixa de ser carregada totalmente)
- `ONFOCUS` – quando a janela passa a ser a janela ativa
- `ONLOAD` – depois que a página é carregada na janela
- `ONUNLOAD` – antes que a página seja substituída por outra ou a janela fechada.

Por exemplo, o código abaixo em uma página carregada em uma janela do browser impedirá que qualquer outra janela esteja ativa até que a janela atual seja fechada. Qualquer tentativa de minimizar a janela, ou de selecionar outra causará o evento tratado por `ONBLUR`, que chamará o método `focus()`, reestabelecendo o estado ativo da janela.

```
<body onblur="focus()"> ... </body>
```

Este outro exemplo, mostra o uso do atributo de evento `ONUNLOAD` para criar uma página que só permite uma única saída, ou seja, só é possível sair da janela atual para entrar em outra definida pelo autor da página. Qualquer tentativa de escolher uma outra URL será sobreposta:

```
<body onunload="location.href='pagina2.html';"> ... </body>
```

Para iniciar um programa, ou rodar uma função, ou executar qualquer procedimento logo depois que *toda a HTML* de uma página tiver sido carregado na janela do browser, pode-se usar o atributo de evento `ONLOAD`:

```
<body onload="iniciarAnimacao()"> ... </body>
```

Todos os atributos de evento também podem ser usados em conjunto:

```
<body onload="iniciar()"
      onunload="parar()"
      onblur="parar()"
      onfocus="iniciar()"
      onerror="location.href=document.location"> ... </body>
```

O manipulador de evento `ONERROR` poderá não funcionar se o erro ocorrer antes que o descritor `<BODY>` que o contém seja carregado.

Comunicação entre janelas

Para passar informações para uma janela recém criada, é necessário obter *uma referência* para a janela. Isto só é possível se a nova janela for criada usando JavaScript. Não funciona para janelas criadas usando HTML. A referência é obtida como valor de retorno do método

`open()`:

```
novaJanela = window.open("pg2.html");
```

Com a referência `novaJanela`, que é *Window*, é possível ter acesso a qualquer propriedade da nova janela e invocar seus métodos, por exemplo:

```
novaJanela.document.write(""); //acrescenta texto à página da janela
novaJanela.focus();           // torna a janela ativa
novaJanela.close();           // fecha a janela
```

Se uma janela é criada usando `open()`, mas o seu valor de retorno não é armazenado em uma variável, não será possível ter acesso às propriedades da *janela filha*. Mas a nova janela sempre pode ter acesso à janela que a criou, manipular suas propriedades e até fechá-la. Toda *janela filha* possui uma propriedade `opener`, que é uma referência à sua *janela mãe*. Para manipular propriedades e invocar métodos ela poderá fazer:

```
opener.focus();                // torna a janela mãe ativa
opener.document.forms[0].elements[2].value = "Oi mãe!";
opener.close();                // mata a mãe
```

É importante verificar que uma propriedade existe, antes de tentar usá-la. Quando se trabalha com múltiplas janelas, é comum uma janela tentar usar uma propriedade que não existe em outra (ou que *ainda* não existe). Se uma página procura um formulário em outra janela e a outra janela não mais apresenta a página que tinha o formulário, o browser acusará um erro, informando a inexistência do objeto.

A tentativa de acessar propriedades inexistentes provoca erros feios em JavaScript. Os browsers mais novos já escondem as janelas de aviso, mas muitos ainda não o fazem. Uma forma de evitá-los é sempre verificar se um objeto está definido, antes de usá-lo. Isto pode ser feito em JavaScript usando a palavra-chave `null`:

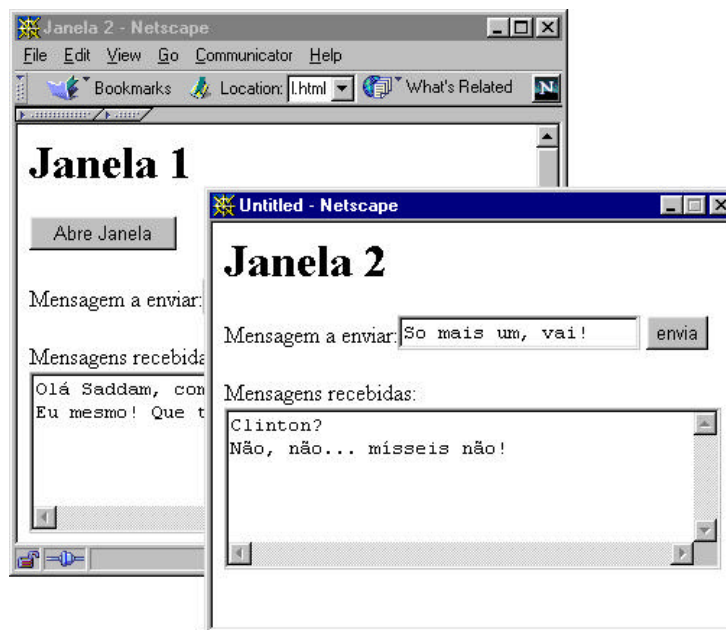
```
if (janela != null) {           // verifica se janela existe
    janela.focus();              // coloca na frente
    if (janela.document.forms[0] != null) { // formulario existe?
        if (campotexto != null) {
            janela.document.forms[0].campotexto.value = "OK";
        }
    }
} else {
    janela = open("pagina.html");
    janela.document.forms[0].elements[0].value = "OK";
}
```

Exercício Resolvido

Monte duas páginas HTML como mostrado na figura abaixo. A primeira página deve ter um botão “Abre Janela” que, quando apertado, deve abrir uma nova janela nas dimensões 360x280 (pixels). Depois de aberta, a nova janela deverá estar na frente da antiga (use `focus()`).

Depois que as duas janelas estiverem abertas, o texto digitado no campo **enviar**, da janela menor, deve aparecer na caixa de mensagens da janela maior, logo que o botão envia for pressionado. Em seguida, a janela maior deverá tornar-se ativa. Pode-se fazer o mesmo na janela maior e passar informações para o campo de mensagens da janela menor.

Use os esqueletos `jan1.html` e `jan2.html` disponíveis no subdiretório `cap5/`. A solução é mostrada a seguir e está nos arquivos `jan1sol.html` e `jan2sol.html`.



Solução

O exemplo a seguir ilustra a comunicação entre janelas. São duas listagens. A primeira é o arquivo para a primeira janela e a segunda o arquivo para a sub-janela. Observe o nome do arquivo “`jan2.html`”. Deve ser idêntico ao primeiro parâmetro do método `open()` na página abaixo.

A página principal contém um botão que permite criar uma nova janela. A partir daí, escreva algo no primeiro campo da nova janela, clique na primeira e veja os dados serem copiados de uma janela para outra.

Este é o código para a janela maior “jan1.html”:

```
<HTML>
<HEAD>
  <TITLE>Janela 2</TITLE>
  <SCRIPT LANGUAGE=JavaScript>

var janela2; // global

function abreJanela() {
  if (janela2 != null) { // janela já está aberta
    janela2.focus();
  } else {
    janela2 = open("jan2.html", "", "height=280,width=360");
  }
}

function envia() {
  janela2.document.f1.mensagens.value += document.f1.enviar.value + "\n";
  document.f1.enviar.value = "";
  janela2.focus();
}
</SCRIPT>
</HEAD>

<BODY>
<H1>Janela 1</H1>

<FORM NAME=f1>
<INPUT TYPE=button VALUE="Abre Janela" ONCLICK="abreJanela()">
<P>Mensagem a enviar:<INPUT TYPE=text NAME="enviar">
<INPUT TYPE=button VALUE="envia" onclick="envia()">
<p>Mensagens recebidas: <br>
<TEXTAREA NAME="mensagens" COLS=40 ROWS=5></TEXTAREA>
</FORM>
</BODY>
```

Este é o arquivo para a janela menor: “jan2.html”

```
<HTML>
<HEAD>
  <TITLE>Janela 2</TITLE>
  <SCRIPT LANGUAGE=JavaScript>
function envia() {
  opener.document.f1.mensagens.value += document.f1.enviar.value + "\n";
  document.f1.enviar.value = "";
  opener.focus();
}
</SCRIPT>
```

```
</HEAD>

<BODY>
<H1>Janela 1</H1>
<FORM NAME=f1>
<P>Mensagem a enviar:<INPUT TYPE=text NAME="enviar">
<INPUT TYPE=button VALUE="envia" onclick="envia()">
<p>Mensagens recebidas: <br>
<TEXTAREA NAME="mensagens" COLS=40 ROWS=5>
</TEXTAREA>
</FORM>

</BODY>
</HTML>
```

Frames HTML

Frames são janelas que estão limitadas dentro de outras janelas. Através de referências especiais, é possível, usando JavaScript, manipular as propriedades de qualquer frame dentro de uma janela ou em outra janela. Antes de apresentar, porém, como é possível manipular frames em JavaScript, vejamos como os frames podem ser construídos em HTML.

Para dividir uma janela em frames, é preciso criar uma página HTML especificando as dimensões relativas ou absolutas das subjanelas em relação à janela que irá conter a página. Uma página de frames não é um documento HTML, pois não contém informação. Todo documento HTML deve ter a forma:

```
<html>
  <head> ... </head>
  <body> ... </body>
</html>
```

O bloco <body> contém a informação da página. O bloco <head>, contém meta-informação, ou seja, informação sobre a página. Páginas de frames têm uma estrutura diferente:

```
<html>
  <head> ... </head>
  <frameset atributos> ... </frameset>
</html>
```

e não podem conter blocos <body>².

² Até podem conter blocos <BODY>, mas isto ora os transforma em páginas de informação, ora não causa efeito algum. Um bloco <BODY> antes do <FRAMESET> faz com que o browser ignore o <FRAMESET>. Um bloco <BODY> após o <FRAMESET> será ignorado por browsers que suportam frames, mas será lido por browsers antigos que não os suportam.

O bloco `<frameset>` define a divisão da janela em linhas (usando o atributo `rows`) ou colunas (usando o atributo `cols`). Os atributos especificam a largura ou altura de cada frame usando valores absolutos, em pixels, ou relativos, em percentagens da largura ou altura da janela principal. Por exemplo, um `<FRAMESET>` da forma (figura ao lado):

```
<FRAMESET COLS="25%,25%,50%"> ... </FRAMESET>
```



divide a janela principal em três colunas, tendo as duas primeiras $\frac{1}{4}$ da largura total, e a última, metade da largura total. De forma semelhante pode-se dividir a janela em linhas. Neste outro exemplo (figura ao lado):

```
<FRAMESET ROWS="100,200,*,100"> ... </FRAMESET>
```



a janela foi dividida em quatro linhas, tendo a primeira e quarta 100 pixels cada de altura, a segunda 200 pixels e a terceira, o espaço restante.

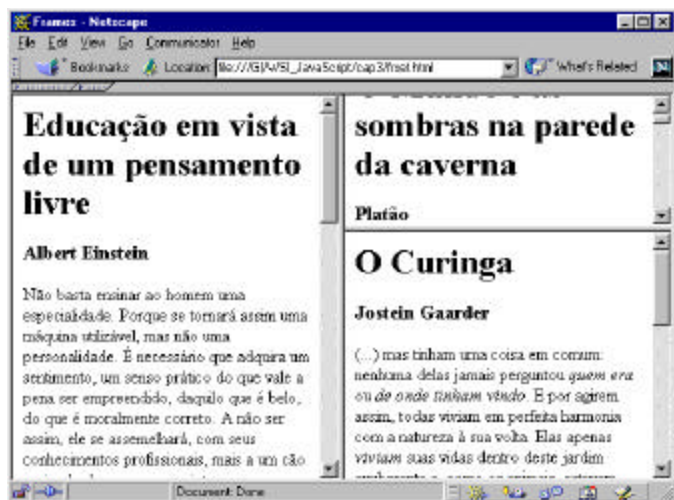
Um bloco `<FRAMESET> ... </FRAMESET>` só pode conter dois tipos de elementos:

- descritores `<FRAME>`, que definem a página HTML que ocupará uma janela. A página HTML poderá ser uma página de informação comum ou outra página de frames que dividirá a sub-janela novamente em linhas ou colunas.
- sub-blocos `<FRAMESET> ... </FRAMESET>` que dividirão outra vez a subjanela (em linhas ou colunas) e poderão conter descritores `<FRAME>` e novos sub-blocos `<FRAMESET>`.

O número de sub-blocos para cada `<FRAMESET>` dependerá do número de linhas (ou colunas) definidas. Para dividir uma janela em linhas e colunas ou de forma irregular, pode-se proceder de duas formas:

- usar um único `<FRAMESET>`, contendo elementos `<FRAME>` que referem-se a páginas de frames (páginas que definem um `<FRAMESET>`), ou
- usar vários `<FRAMESET>` em cascata na mesma página.

Usaremos as duas formas para montar a janela ao lado. Na primeira versão, utilizaremos *dois* arquivos de frames: `frset1.html` dividirá a janela principal em duas colunas, e `frset2.html` dividirá a segunda coluna em duas linhas. Na segunda versão, precisaremos de apenas *um* arquivo de frames (`frset.html`). As duas versões utilizarão três arquivos de



informação: `um.html`, `dois.html` e `tres.html`. O resultado final é o mesmo, mas as duas formas podem ser manipuladas de forma diferente em JavaScript.

Na primeira versão temos dois arquivos. Os trechos em **negrito** indicam as ligações entre eles. O primeiro é `frset1.html`, que referencia uma página de informação:

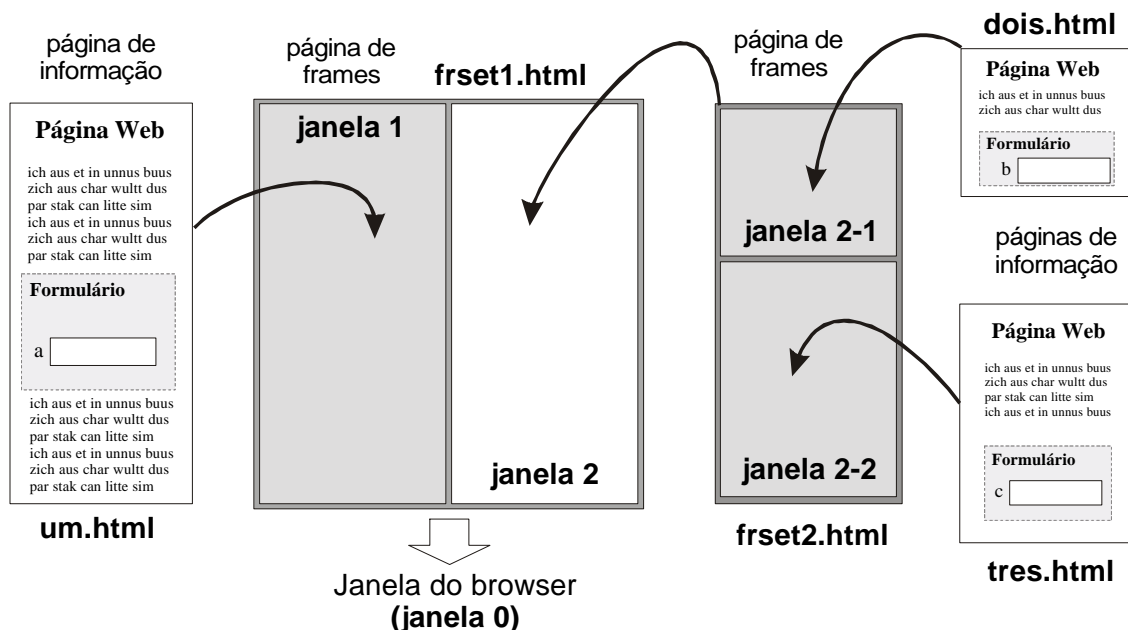
```
<html>
<head> ... </head>

<frameset cols="50%,50%">
  <frame name="janela1" src="um.html">
    <frame name="janela2" src="frset2.html">
</frameset>
</html>
```

e chama `frset2.html`, com mais duas páginas de informação, listado abaixo:

```
<html>
<head> ... </head>
<frameset rows="35%,65%">
  <frame name="janela2_1" src="dois.html">
  <frame name="janela2_2" src="tres.html">
</frameset>
</html>
```

A figura abaixo mostra a organização das páginas de informação e das páginas de frames na janela do browser.



Observe que há três níveis de páginas. No nível mais alto está a página `frset1.html`, que ocupa toda a janela do browser. No segundo nível estão os arquivos `um.html` e `frset2.html`. E no terceiro nível, encontramos os arquivos `dois.html` e `tres.html`.

Na segunda versão, temos apenas um arquivo de frames contendo referências para os três arquivos de informação. Em negrito está mostrado o segundo *frame-set*:

```
<html>
<head> ... </head>
<frameset cols="50%,50%">
  <frame name="janela1" src="um.html">
    <bframeset rows="35%,65%">
      <frame name="janela2_1" src="dois.html">
      <frame name="janela2_2" src="tres.html">
    </bframeset>
  </frameset>
</html>
```

Esta segunda versão, possui apenas dois níveis. No primeiro, a página de frames `frset.html`, no segundo, as páginas de informação. A aparência final é a mesma, nas duas versões, mas na primeira versão há uma janela a mais (`janela2`) que pode ser manipulada em JavaScript e em HTML. Se a `janela2` for utilizada como alvo de um link HTML:

```
<a href="pagina.html" TARGET="janela2"> link </A>
```

os frames `janela2_1` e `janela2_2`, que estão em um nível abaixo de `janela2` deixarão de existir e `pagina.html` ocupará toda a segunda coluna da janela do browser. Isto não poderá ser feito na segunda versão, pois ela só possui dois níveis.

Se o link estiver dentro da página `dois.html` ou `tres.html`, a sintaxe abaixo, usando o nome especial `_parent` causará um resultado equivalente:

```
<a href="pagina.html" TARGET="_parent"> link </A>
```

Usando frames em JavaScript

Em JavaScript, frames podem ser manipulados por referências (objetos) que indicam relações hierárquicas, posição dos frames ou seus nomes. Toda página de frames possui um vetor `frames` que contém referências para os frames, na ordem em que aparecem no `<FRAMESET>`.

Suponha a seguinte estrutura de frames

```
<html>
<head> ... </head>
<frameset cols="50%,50%">
  <frame name="janela1" src="um.html">                                <!-- frames[0] -->
  <frameset rows="35%,65%">
    <frame name="janela2_1" src="dois.html">                          <!-- frames[1] -->
    <frame name="janela2_2" src="tres.html">                          <!-- frames[2] -->
  </frameset>
</frameset>
</html>
```

Um script nesta página pode manipular os seus frames de duas formas: pelo nome ou através do vetor `frames`. O código abaixo mostra duas maneiras diferentes de mudar a cor de fundo das páginas do primeiro e do último frame:

```
frames[0].document.bgColor = "red";
frames[2].document.bgColor = "blue"; // ... é a mesma coisa que...
janela1.document.bgColor = "red";
janela2_2.document.bgColor = "blue";
```

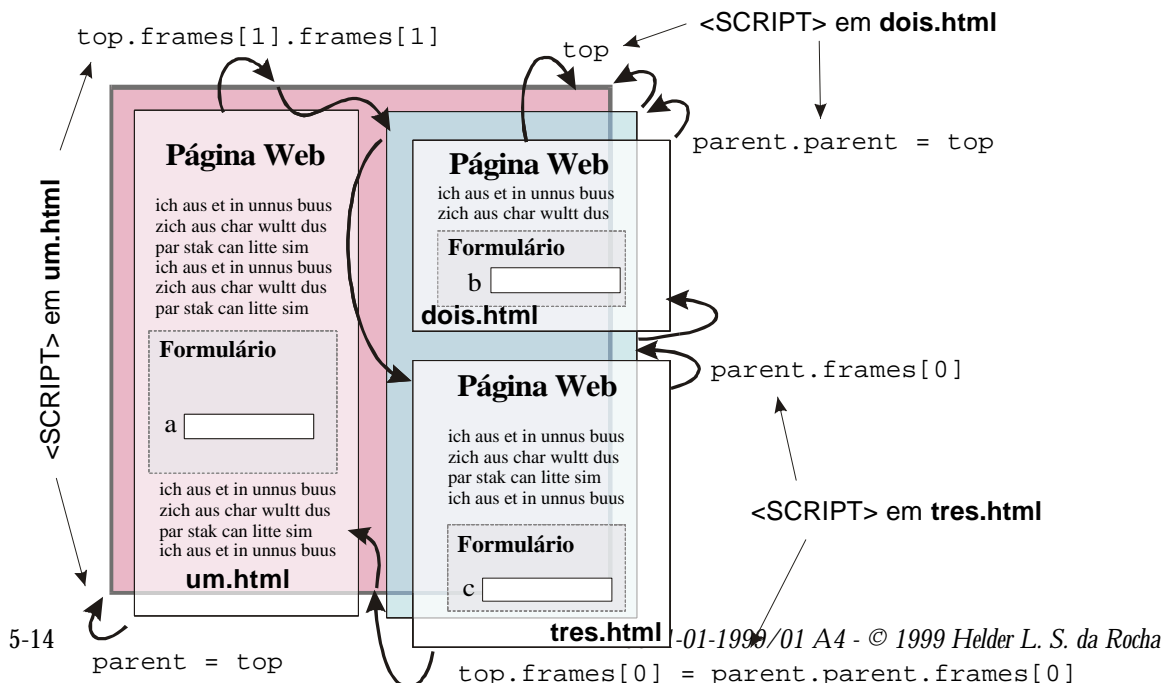
Geralmente não há informação alguma nas páginas de frames, muito menos scripts. O mais comum é existirem scripts nas páginas de informação contidas nos frames. Sendo assim, é necessário haver uma referência para a página que contém o frame. Em JavaScript, esta referência é a propriedade `parent`. Para mudar a cor da página do primeiro frame a partir de um script rodando no último, pode-se fazer:

```
parent.frames[0].document.bgColor = "red"; // ... ou
parent.janela1.document.bgColor = "red";
```

Isto funciona porque `parent` é *Window*, possui a propriedade `frames`, e conhece o nome `janela1`, que está definido no código HTML da página que contém. O código acima não funcionaria se tivéssemos usado a estrutura de frames com três níveis, como o primeiro exemplo da seção anterior. Para ter acesso ao primeiro frame, teríamos que subir dois níveis, até o nível mais alto, para então descer um nível até `frames[0]`. Poderíamos usar `parent` duas vezes ou a propriedade `top`, que representa o nível mais alto:

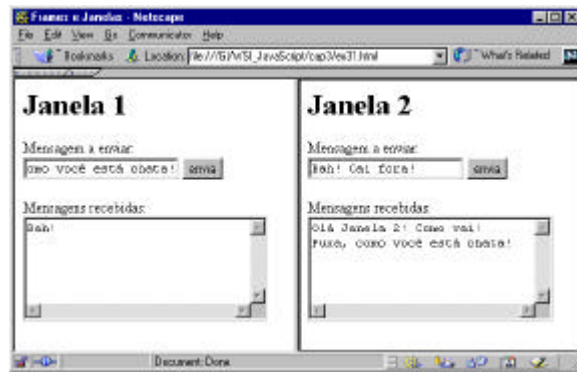
```
parent.parent.frames[0].document.bgColor = "red";
top.janela1.document.bgColor = "red";
```

A partir de `top` pode-se chegar a qualquer frame, usando seu nome ou o vetor `frames`. Nos casos onde existem apenas dois níveis de frames, `top` é sinônimo de `parent`. A figura abaixo mostra várias formas de comunicação entre frames:

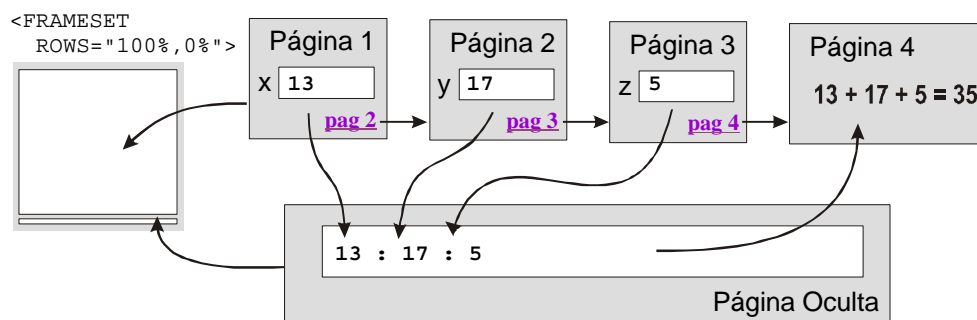


Exercícios

- 5.1 Repita o exercício resolvido deste capítulo criando uma página de frames e posicionando as duas janelas nesta estrutura. Altere as páginas de forma que elas possam trocar valores entre frames (veja a figura ao lado).



- 5.2 Este exercício usa frames para passar informações entre páginas. Divida a janela em dois frames, sendo um frame fixo, com altura zero (deverá ficar escondido na parte de baixo da página) e outro, ocupando toda a página. Crie uma página HTML contendo apenas um formulário e um elemento `<textarea>`. Crie mais quatro páginas HTML. A primeira delas deverá ser carregada no frame maior. As três primeiras são idênticas e deverão ter, cada uma, uma caixa de texto, onde o usuário deverá digitar um número, e um link, para a página seguinte. Quando o usuário decidir seguir para a página seguinte, o texto digitado deverá ser copiado para o `<textarea>` da página escondida. Ao chegar na quarta página, esta deverá exibir os números digitados em cada página e a sua soma. (em vez de `<textarea>`, pode-se usar `<input type=hidden>`, e manter os dados temporários invisíveis. Veja o diagrama da aplicação na figura abaixo. Esta é uma forma de passar informações entre páginas sem usar *cookies*.



- 5.3 Use `setTimeout()` e o tipo `Date` para implementar um relógio como o mostrado na figura abaixo. O relógio deverá ser iniciado logo que a página for carregada e atualizado a cada segundo. Implemente um mecanismo para recarregar a página caso o dia mude (use `location.reload()`).

