12

JavaScript e Java

APPLETS JAVA OFERECEM RECURSOS QUE VÃO MUITO ALÉM do que se dispõe com JavaScript e HTML. Por outro lado, applets pouco interagem com a página. Não podem alterar propriedades da página nem utilizar formulários HTML. JavaScript oferece recursos de programação e integração total com o browser e a página, mas não pode ir além das limitações do HTML e do browser. Usando Java e JavaScript juntos, une-se a riqueza de recursos de Java à integração do JavaScript com o browser o que permite permite explorar o melhor de cada tecnologia em uma mesma aplicação.

Os browsers mais populares suportam a o controle de applets a partir de JavaScript e vice-versa. Isto inclui os browsers Netscape Navigator a partir da versão 3.0 e o browser Microsoft Internet Explorer a partir da versão 4.0[1]. Neste capítulo, mostraremos como manipular com os applets em uma página Web, e exploraremos, com exemplos e exercícios resolvidos, a comunicação entre applets e JavaScript.

Applets Java

Applets são pequenas aplicações geralmente escritas em Java que são executadas pelo browser. Diferentemente do que ocorre com JavaScript, o código Java não é interpretado pelo browser. Um applet também não tem código Java que o browser possa interpretar, já que foi compilado para uma linguagem de máquina. Browsers que suportam Java possuem uma plataforma virtual, a "Java Virtual Machine", que é capaz de interpretar a linguagem de máquina Java, chamada de *bytecode*.

Applets podem ser usados para desenvolver aplicações que seriam impossíveis em JavaScript por causa das limitações do HTML, do protocolo HTTP e do próprio browser. Com um applet, é possível estender um browser fazendo-o suportar, por exemplo, novos

[1] Além da comunicação entre applets e scripts, o Netscape Navigator, permite ainda que o programador utilize diretamente classes da API Java, chame seus métodos e crie objetos Java a partir de instruções JavaScript. Não discutiremos este recurso aqui por ele não ter suporte além dos browsers Netscape.

protocolos de comunicação e segurança, novos formatos de mídia, etc. O preço dessa liberdade é sua fraca integração com o HTML da página. Aplicações Web baseadas em Java pouco ou nada aproveitam do HTML da página a não ser um espaço gráfico para sua exibição. Com JavaScript, é possível aumentar essa integração.

Applets são aplicações gráficas e precisam de uma página HTML para poderem executar. São exibidos na página de forma semelhante a imagens: carregam um arquivo externo, ocupam um espaço com determinada altura e largura, e podem ter seu alinhamento em relação ao texto definido pelos mesmos atributos presentes em `<IMG>`. A sintaxe do elemento HTML `<APPLET>` está mostrada abaixo. Tudo o que não estiver em negrito é opcional:

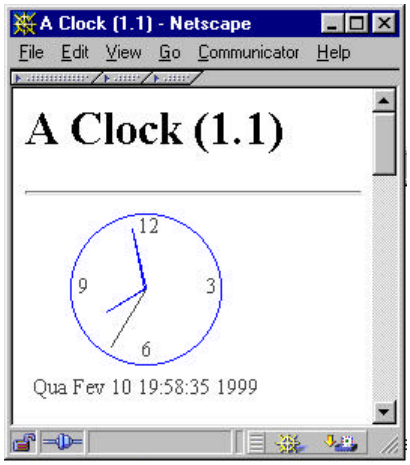
```
<APPLET
   CODE="nome_do_programa.class"
   HEIGHT="altura em pixels"
   WIDTH="largura em pixels"
   NAME="nome_do_objeto"
   CODEBASE="diretório base do arquivo de classe Java"
   ALT="texto descritivo"
   HSPACE="margens externas laterais em pixels"
   VSPACE="margens externas verticais em pixels"
   ALIGN="left" ou "right" ou "top" ou "middle" ou "bottom" ou
         "texttop" ou "absmiddle" ou "absbottom" ou "baseline"
   MAYSCRIPT>
      <PARAM NAME="nome" VALUE="valor">
         ...
      <PARAM NAME="nome" VALUE="valor">
</APPLET>
```

Diferentemente de `<IMG>`, o elemento `<APPLET>` é um bloco e possui um descritor de fechamento `</APPLET>`. Entre `<APPLET>` e `</APPLET>` pode haver nenhum ou vários elementos `<PARAM>`, que contém parâmetros necessários ou não (depende do applet) para o funcionamento da aplicação. Cada elemento `<PARAM>` contém um par de atributos obrigatórios. O valor do atributo `NAME` é definido pelo programador do applet. Através dele, o programa pode recuperar um valor que o autor da página (que não precisa saber Java) definiu no atributo `VALUE`.

O atributo `MAYSCRIPT` é necessário se o applet pretende ter acesso ao código JavaScript da página. Sem este atributo, qualquer tentativa do applet de acessar variáveis ou executar funções ou métodos JavaScript causará em uma exceção de segurança no applet.

Existem muitos applets úteis disponíveis gratuitamente na Web que podem ser usados por autores de páginas Web e programadores JavaScript sem que precisem saber Java. Os mais populares implementam banners para rolagem de texto, ícones inteligentes, gráficos, planilhas de dados e interfaces para bancos de dados. A maioria são configuráveis através de

parâmetros que o autor da página define em elementos <PARAM>. No apêndice A há uma lista de sites onde se pode encontrar tais applets.



O exemplo a seguir mostra como incluir o applet `Clock2` em uma página Web. Este applet é distribuído pela *Sun* gratuitamente em `http://java.sun.com` e juntamente com o ambiente de desenvolvimento Java. O applet pode ser incluído na página da forma *default*, sem especificar parâmetros, ou definindo parâmetros que alteram a cor de fundo, dos ponteiros e do mostrador.

O applet é distribuído com uma página HTML que mostra como usá-lo. Ele deve ser incluído na página HTML usando o nome do arquivo executável java, que é `Clock2.class` e deve ocupar uma área de no mínimo 170x150 pixels:

```
<applet code="Clock2.class" height=150 width=170></applet>
```

Com o código acima, o relógio aparece na página como mostrado na figura, com ponteiros azuis, visor com letras pretas e fundo branco. O autor do applet permite, porém, que o autor da página altere esses parâmetros através de descritores <PARAM>. Os três parâmetros modificáveis são:

- `bgcolor` – cor de fundo (branco é *default*)
- `fgcolor1` – cor dos ponteiros e dial (azul é *default*)
- `fgcolor2` – cor dos números e ponteiro de seguntos (preto é *default*)

Todos os parâmetros devem receber como valor um número hexadecimal representando uma cor no formato RGB (mesmo formato usado em HTML): `ff0000` – vermelho, `ffffff` – branco, `0000ff` – azul, etc.

Portanto, para incluir o relógio acima em uma página, com um fundo cinza claro, ponteiros marrons e letras douradas, o código seria:

```
<applet code="Clock2.class" width=170 height=150>
      <param name=bgcolor value="dddddd">
      <param name=fgcolor1 value="800000">
      <param name=fgcolor2 value="808000">
</applet>
```

Caso o applet esteja em um diretório diferente daquele onde está a página, será necessário usar o atributo `CODEBASE`, para informar a URL base. Por exemplo, se o arquivo `.class` que usamos acima estiver em `http://www.abc.com/clocks/`, precisamos usar:

```
<applet codebase="http://www.abc.com/clocks/" code="Clock2.class" ... >
   ...
</applet>
```

Objeto Applet

O tipo de objeto *Applet* representa, no modelo de objetos JavaScript, um applet Java embutido em uma página Web. Tendo uma referência para um objeto *Applet*, o programador pode controlar um applet Java usando JavaScript, *sem que precise ter acesso ao código* do applet. Precisará apenas saber os nomes dos métodos públicos do applet para que possa invocá-los via JavaScript. É possível também fazer o inverso: controlar JavaScript a partir de applets. Neste caso, *é preciso ter acesso ao código* do applet e conhecer a linguagem Java.

Não é possível criar objetos *Applet* usando JavaScript, apenas. Objetos *Applet* são fornecidos pelo código HTML da página. Se houver na página um bloco `<APPLET>` que tenha carregado um arquivo executável Java, existe um objeto *Applet* utilizável em JavaScript.

Uma página pode ter vários applets. Eles podem ser obtidos através da propriedade `document.applets` – um vetor que, como `document.images` e `document.forms`, contém referências para todos os applets presentes na página, na ordem em que aparecem no código. Por exemplo, em uma página com três applets, o primeiro e terceiro podem ser referenciados da forma:

```
appy1 = document.applets[0];   // primeiro applet da página atual
appy3 = document.applets[2];   // terceiro applet da página atual
```

Os applets de uma página também são são acessíveis através de um nome, especificado pelo atributo HTML opcional `NAME`. Acessar um applet pelo nome é mais prático e evita que a modificação da ordem dos applets na página afete o funcionamento da aplicação. Por exemplo, o applet:

```
<applet code="Clock2.class" name="reloj" width=170 height=150>
</applet>
```

pode ser referenciado em qualquer atributo de eventos ou bloco `<SCRIPT>` da página da forma:

```
appy = document.reloj;   // Applet!
```

O número de applets em uma página pode ser descoberto através da propriedade `applets.length`, de `document`:

```
numApplets = document.applets.length;
```

As propriedades dos objetos *Applet* são todas as variáveis públicas (definidas no código Java) do applet. As propriedades do HTML (`code`, `name`, `height`, `width`) podem ser lidas somente no browser Microsoft Internet Explorer. No Netscape, elas não existem.

Os métodos dos objetos *Applet* consistem de todos os métodos públicos (definidos no código Java) do applet. Não há eventos JavaScript associados ao objeto *Applet*.

JS11-01-1999/01 A4 - © 1999 Helder L. S. da Rocha

Controle de Applets via JavaScript

O controle de applets a partir do código JavaScript é bastante simples, pois em muitas aplicações não exige conhecimentos de Java nem dos detalhes internos do applet. Conhecendo-se os métodos e variáveis públicas de um applet, pode-se acessá-los diretamente pelo nome através do objeto, que é referência ao applet em JavaScript[2].

No ambiente de desenvolvimento Java (JDK – Java Development Kit), há uma ferramenta chamada **`javap`** que imprime uma lista das *assinaturas* de todos os métodos e variáveis públicas de um programa Java compilado (arquivo com extensão `.class`). A assinatura consiste do nome do método, seu tipo de retorno e os tipos de seus argumentos. Por exemplo, suponha que você possua um arquivo `Carta.class`, que é um applet Java e está incluído em uma página HTML através do bloco:

```
<applet code="Carta.class" height=100 width=200 name="mens">
</applet>
```

Você não conhece o formato e nome dos métodos de `Carta.class` mas possui o JDK, que tem a ferramenta `javap`. Rodando a ferramenta `javap` sobre o arquivo `Carta.class`, obtém-se o seguinte:

```
c:\> javap Carta                    (é preciso omitir a extensão .class)
public class Carta extends java.applet.Applet {
  public int numero;
  public void mudarMensagem(String);
  public String lerMensagem();
}
C:\>_
```

A primeira linha, identifica a *classe java* (`Carta`), que é um applet. Todo programa em Java é considerado uma *classe*. A segunda linha contém a declaração de uma variável chamada `numero`. A palavra `public` indica que se trata de uma variável pública (pode ser usada em JavaScript) e a palavra `int` indica que é um número inteiro. Se formos atribuir um valor à variável `numero`, através de JavaScript, precisaremos ter o cuidado de passar um número inteiro e não um *String* ou outro tipo de dados. Java, diferente de JavaScript, só permite que uma variável receba um valor, se for de um tipo previamente declarado para a variável.

As duas últimas linhas contém as assinaturas dos métodos `mudarMensagem()` e `lerMensagem()`. A palavra `public` indica que ambos são públicos e portanto podem ser usados em JavaScript. O método `mudarMensagem()` é declarado `void`, o que significa que ele não retorna valor. Ele recebe como argumento uma variável que deve necessariamente

[2] Variáveis e métodos em Java que são declarados 'static' não são acessíveis através da referência da applet mas através do tipo *Applet*, da forma Applet.variavel ou Applet.metodo().

ser um objeto do tipo *String*. O método `lerMensagem()` não tem argumentos, mas retorna um valor do tipo *String*.

Com estas informações, temos condições de manipular as variáveis e métodos do applet definido pela classe `Carta.class`, através de propriedades e métodos de um objeto *Applet*:

```
appy = document.applets[0];    // se for o primeiro applet
appy.numero = 6;
document.write("A mensagem atual é " + appy.lerMensagem());
appy.mudarMensagem("Esta é a nova mensagem!");
document.write("A mensagem agora é " + appy.lerMensagem());
document.write("O número é " + appy.numero);
```

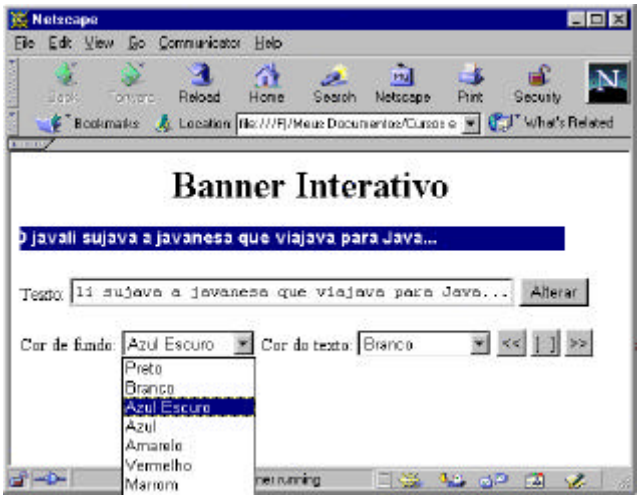
No exercício resolvido a seguir, mostraremos um exemplo prático do controle de applets através de JavaScript.

Exercício Resolvido

Para este exercício, utilize o arquivo `Banner.class`, que é um applet que faz uma mensagem de texto rolar horizontalmente na tela. A mensagem do `Banner` pode ser definida na página HTML dentro de um atributo `<PARAM>` com o nome (atributo `NAME`) "`MSG`". O texto que estiver no campo `VALUE` será utilizado como a mensagem a ser exibida. Se o `<PARAM>` não estiver presente, o applet ainda funcionará, porém, apresentará uma mensagem *default*.

`Banner` possui vários métodos públicos que permitem mudar a mensagem atual, mudar as cores de fundo e do texto, parar a rolagem e aumentar a velocidade de rolagem para a direita ou para a esquerda. Os métodos públicos de `Banner` são os seguintes:

```
public void corDeFundo(int r, int g, int b) {
public void corDoTexto(int r, int g, int b) {
public void mensagem(String msg) {
public void esquerda() {
public void direita() {
public void para() {
```



Use a página esqueleto `Banner.html` (figura ao lado) para:

a) acrescentar o applet no lugar indicado com o texto inicial "Bom Dia!".

b) programar em JavaScript o campo de texto para que ele mude a mensagem do applet.

c) programar os campos `<SELECT>` para

que mudem as cores de fundo e do texto.

d) programar os botões para que invoquem métodos que façam o texto parar (botão "`[ ]`"), andar mais rápido para a esquerda (botão "<<"), e para a direita (botão ">>") .

Solução

A primeira tarefa é colocar o applet na página. Definimos o parâmetro `MSG` com o valor `"Bom Dia!"`, como foi pedido no requisito (a):

```
<!—Coloque o applet aqui -->
<body>
<h1>Applets controlados por JavaScript</h1>
<applet code="Banner.class" height=20 width=450 hspace=10>
   <param name="msg" value="Bom Dia!">
</applet>
<form>
(...)
```

Com o bloco de código acima, o applet já deve aparecer na página e começar a rolar para a esquerda. Para permitir a mudança do texto durante a execução do applet, chamamos o método `mensagem()`, que muda o texto para o string recebido. O string é obtido do campo de textos `novotexto`::

```
<p>Texto:
<input type=text name=novotexto size=45>
<input type=button value="Alterar"
       onclick="document.applets[0].mensagem(this.form.novotexto.value)">
```

A mudança das cores exige mais trabalho já que os métodos `corDeFundo()` e `corDoTexto()` recebem três parâmetros inteiros, e a lista `<SELECT>` fornece apenas um valor *String*, com os valores RGB separados por vírgulas:

```
<option value="255,255,0">Amarelo</option>
```

Criamos então, uma função `cor()`, que converte o valor da opção selecionada em três números inteiros. A função, que recebe um objeto *Select* como argumento, também identifica qual das duas listas foi selecionada, para invocar o método correto:

```
<head>
<script>
function cor(selObj) {
   corStr = selObj.options[selObj.selectedIndex].value;
   rgb = corStr.split(",");
   r = parseInt(rgb[0]);
   g = parseInt(rgb[1]);
   b = parseInt(rgb[2]);
   if (selObj.name == "bg") {
      document.applets[0].corDeFundo(r, g, b);
```

```
      } else if (selObj.name == "fg") {
         document.applets[0].corDoTexto(r, g, b);
      }
   }
   </script>
   </head>
```

A única alteração necessária nos blocos `<SELECT>` a programação do atributo `ONCHANGE`, para chamar a função, passando o próprio *Select* como argumento:

```
<p>Cor de fundo:
<select name=bg onchange="cor(this)">
   <option value="0,0,0">Preto</option>
   <option value="255,255,255">Branco</option>
   <option value="0,0,128">Azul Escuro</option>
   <option value="0,0,255" selected>Azul</option>
   <option value="255,255,0">Amarelo</option>
   <option value="255,0,0">Vermelho</option>
   <option value="128,0,0">Marrom</option>
   <option value="0,128,0">Verde Escuro</option>
</select>
(...)
```

A programação dos botões é simples. Eles simplesmente chamam o método correspondente à sua função:

```
<input type=button value="&lt;&lt;"
       onclick="document.applets[0].esquerda()">
<input type=button value="[  ]"
       onclick="document.applets[0].para()">
<input type=button value="&gt;&gt;"
       onclick="document.applets[0].direita()">
</form>
</body>
```

A listagem completa desta solução está no arquivo `Bannersol.html`.

Exercícios

12.1 Se o applet `Desenho.class` (disponível no diretório `cap12/`) for instalado em uma página HTML, a página passará a ter uma área onde o usuário poderá fazer desenhos em preto-e-branco. Há, porém, dois métodos públicos no applet: `mudaCor()` e `clear()` que permitem respectivamente mudar a cor e limpar a tela. Inclua o applet `Desenho` na página `Desenho.html` (figura ao lado) e programe: a) o

botão, para que limpe a tela de desenho ao ser apertado, e b) a lista de opções, para que mude a cor do lápis de acordo com a seleção do usuário.

Controle de JavaScript através de Applets

Para fazer applets controlarem páginas HTML e código JavaScript, é preciso programar os applets em Java. Esta seção, portanto, assume que o leitor conhece e está familiarizado com a linguagem Java. Para poder testar os exemplos apresentados e resolver os exercícios, é necessário ter um ambiente de desenvolvimento Java instalado, como o JDK da Sun.

Um applet Java só poderá ter acesso a uma página HTML se o autor da página permitir. A permissáo é dada colocando o atributo `MAYSCRIPT` em cada applet que tem permissão para acessar a página. `MAYSCRIPT` só é necessário para que applets acessem scripts e não o contrário, como nos exemplos que vimos até aqui.

O suporte à JavaScript em aplicações escritas em Java é obtido através do pacote `netscape.javascript`, fornecido pela Netscape. Este pacote é suportado não só nos browsers da Netscape mas também nos browsers Microsoft Internet Explorer (a partir da versão 4).

O pacote contém a classe `JSObject` e a exceção `JSException`. `JSObject` é usado para representar qualquer objeto JavaScript em Java. Toda aplicação Java que pretenda se comunicar com uma página HTML via JavaScript precisa usá-lo. A melhor forma de ter acesso aos recursos disponíveis em todo o pacote é importá-lo no código da aplicação:

```
import netscape.javascript.*;
```

A maior parte dos métodos públicos usados na comunicação Java com JavaScript estão na classe `JSObject`, listados nas tabelas abaixo (*Atenção*: estes métodos são métodos Java. Não os confunda com métodos JavaScript). `JSObject` contém um único método estático, que retorna uma referência à janela do browser:

Assinatura do método de classe (public static)	**Descrição**
`JSObject` **`getWindow`**`(Applet` ***`applet`***`)`	Obtém um `JSObject` representando a janela do browser onde está o ***`applet`*** passado como argumento.

Para manipular com os objetos do browser, é preciso obter primeiro uma referência para a janela do browser. Para isto, utilizamos o método `getWindow()` passando o applet atual como argumento, no código Java, da forma:

```
JSObject janela = JSObject.getWindow(this);       // código Java!
```

Depois de obtida uma referência a um objeto JavaScript, podemos obter referências a todas as suas propriedades e métodos usando os *métodos de instância* da classe `JSObject`, listados na tabela abaixo. Esses métodos precisam ser invocados sobre objetos `JSObject`. Todos os métodos que retornam valor, retornam `Object` (exceto `toString()`). Se houver

necessidade de usar tipos primitivos em parâmetros eles precisam ser encapsulados em objetos como `Integer`, `Boolean`, etc.:

Assinatura do método de instância (public)	**Descrição**
`Object` **`getMember`**`(String` *`nome`*`)`	Recupera uma propriedade de um objeto JavaScript pelo *`nome`*.
`Object` **`getSlot`**`(int` *`indice`*`)`	Recupera uma propriedade de um objeto JavaScript pelo *`índice`* .
`Object` **`eval`**`(String` *`expressao`*`)`	Executa expressões JavaScript.
`Object` **`call`**`(String` *`nomeMetodo`*`, Object[]` *`args`*`)`	Chama um método ou função JavaScript. Os argumentos da função devem ser passados no vetor *`args`*.
`void` **`setMember`**`(String` *`nome`*`, Object` *`valor`*`)`	Define um novo *`valor`* para uma propriedade de um objeto JavaScript pelo *`nome`*.
`public void` **`setSlot`**`(int` *`indice`*`, Object` *`valor`*`)`	Define um novo *`valor`* para uma propriedade de um objeto JavaScript pelo *`índice`*.
`void` **`removeMember`**`(String` *`nome`*`)`	Remove uma propriedade de um objeto JavaScript pelo *`nome`*.
`String` **`toString`**`()`	Devolve uma String com uma descrição do objeto.

Qualquer elemento HTML referenciado na hierarquia de objetos JavaScript pode ser obtido a partir da referência à janela do browser. O método `getMember()`, invocado em qualquer `JSObject`, retorna uma referência Java para a o nome da propriedade passada como parâmetro ou `null` se a propriedade não existir. Se a propriedade for um vetor em JavaScript, cada elemento pode ser recuperado com o método `getSlot()`, passando o índice correspondente como argumento. Como exemplo do uso desses métodos, considere o seguinte trecho HTML:

```
<body>
 <form>
   <input type=text name=dados>
 </form>
</body>
```

As seguintes operações, dentro do codigo do applet (Java), permitem que ele tenha acesso ao conteúdo de um campo de textos na página HTML:

```
JSObject janela = JSObject.getWindow(this);      // código Java!
JSObject docmt = (JSObject)janela.getMember("document");
JSObject frmArray = (JSObject)docmt.getMember("forms");
```

```
JSObject form1 = (JSObject) frmArray.getSlot(0);
JSObject campoTxt = (JSObject)form1.getMember("dados");
String valor = (String)campoTxt.getMember("value");
```

Expressões em JavaScript podem ser executadas a partir de applets Java usando o método `eval()`. É semelhante à função `eval()` do JavaScript só que neste caso, é um método Java, que opera sobre um `JSObject`. A obtenção da janela do browser é o suficiente para usar `eval()`. Com `eval()`, applets poderão ter diálogos modais: applets não possuem janelas de diálogo pré-definidas como as janelas de alerta e confirmação do JavaScript. Essas janelas, disponíveis em JavaScript, podem ser usadas em Java com `eval()`:

```
JSObject.getWindow().eval("alert(\"Saudações Javanesas!\"");
```

O método `eval()` também pode ser usado para obter referências a propriedades de *Window* e objetos de sua hierarquia de forma mais direta que usando sucessivas chamadas a `getMember()` e `getSlot()`. O código abaixo tem o mesmo efeito que o listado anteriormente para acessar o valor de um campo de texto:

```
JSObject janela = JSObject.getWindow(this);
String valor = (String)janela.eval("document.form1.campo.value");
```

Outra forma de chamar métodos ou funções JavaScript a partir de Java é usando o método `call()`. Este método recebe dois argumentos: o primeiro é o nome da função ou método, o segundo, os argumentos que esta função ou método recebem, dentro de um vetor. O vetor pode ser de qualquer descendente de `java.lang.Object`. Para passar argumentos de tipos primitivos, é necessário empacotá-los em objetos como `Integer`, `Boolean`, etc.

O exemplo abaixo é semelhante ao mostrado anteriormente com `eval()`. Desta vez usamos `call()`:

```
JSObject win = JSObject.getWindow();
String[] args = {"Saudações Javanesas!"};
win.call("alert", args);
```

Neste outro trecho de código, chamamos uma função `soma()`, disponível na página JavaScript, que recebe dois inteiros (os números 7 e 9) e retorna a soma. Precisamos colocar valores do tipo `int` dentro de objetos `Integer`:

```
JSObject win = JSObject.getWindow();
Object[] args = {new Integer(7), new Integer(9)};
Integer intObj = (Integer)win.call("soma", args);
int resultado = intObj.intValue();
```

Os métodos `setMember()` e `setSlot()` são usados para definir novos valores ou novas propriedades em JavaScript a partir de Java. Por exemplo, para definir a propriedade value de um campo de textos, pode-se usar:

```
JSObject win = JSObject.getWindow();
JSObject campoTxt = (JSObject)win.eval("document.form1.dados");
campoTxt.setMember("value", "Contatos Imediatos de JavaScript!");
```

Na seção seguinte, resolveremos um exercício que utiliza comunicação Java-JavaScript nos dois sentidos.

Exercício Resolvido

O objetivo deste exercício é estender a aplicação proposta no exercício 12.1 para que um evento ocorrido no applet Java provoque uma alteração na página HTML. Faz parte deste exercício a alteração de um programa em Java, portanto é necessário que esteja disponível um ambiente de desenvolvimento Java como o JDK da Sun. A figura abaixo mostra a aplicação rodando no Microsoft Internet Explorer.
Utilize os seguintes arquivos, localizados no diretório `cap12/`:

- `Desenha2.java`
- `Desenha2.html`

As etapas do exercício são as seguintes:

a) Primeiro altere o arquivo `Desenha2.html`, criando uma função que receba dois argumentos do tipo *String* e preencha os campos de texto `coordx` e `coordy` com os valores recebidos.

b) Depois altere o programa `Desenha2.java` para que na ocorrência do evento de movimento do mouse (método `mouseMoved()`) as coordenadas `x` e `y` do ponteiro do mouse sejam recuperadas e que a função JavaScript definida no item anterior seja chamada com os valores.

c) Compile o programa, teste e carregue no browser.

Solução

A primeira parte é simples. Consiste apenas em definir a função que irá alterar os valores dos campos de texto. Esta função será chamada a partir do applet Java.

```
function setCoords(x, y) {
   document.forms[0].coordx.value = x;
   document.forms[0].coordy.value = y;
}
```

e irá alterar os valores dos campos de texto `coordx` e `coordy`, definidos no formulário:

```
(...)
x <input type=text name=coordx size=4><br>
y <input type=text name=coordy size=4>
</form>
(...)
```

Agora precisamos alterar o programa em Java. Uma listagem parcial do programa está mostrada abaixo. Os trechos que foram adicionados ao programa original estão em negrito:

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import netscape.javascript.*;  // suporte a JavaScript pelo Applet

public class Desenha2 extends Applet
                      implements MouseMotionListener,
                                 MouseListener        {
   private Dimension dim;
   private int x, y, oldx, oldy;
   private boolean clearAll = false;
   private Color cor = Color.black;
   private JSObject win;       // janela do browser
   private String[] args;      // argumentos da função setCoords()

   public void init() {
      dim = getSize();
      this.addMouseMotionListener(this);
      this.addMouseListener(this);
      win = JSObject.getWindow(this);
      args = new String[2];    // são 2 os argumentos da função
   }

   public void mouseDragged(MouseEvent e {
      if ((x == 0) && (y == 0)) {
        x = e.getX();
        y = e.getY();
      }
      oldx = x;
      oldy = y;
      x = e.getX();
      y = e.getY();
      args[0] = "" + x;
      args[1] = "" + y;
      win.call("setCoords", args);
      repaint();
   }

   public void mouseMoved(MouseEvent e) {
```

```
            args[0] = "" + e.getX();       // String com o valor de x
            args[1] = "" + e.getY();       // String com o valor de y
            win.call("setCoords", args);   // Chama função setCoords(x, y)
        }

        // (... restante do código inalterado: métodos não
        // mostrados aqui nao foram modificados ... )
    }
```

Para compilar o programa[3], precisamos definir o caminho onde o compilador poderá procurar pelo pacote `netscape.javascript`. Isto é feito através da variável de ambiente `CLASSPATH` e só é necessário para a compilação, portanto, podemos defini-la na linha de comando.

A localização do pacote `netscape.javascript` depende do browser instalado e da plataforma. Se o seu browser é Microsoft Internet Explorer em Windows95/98, ele está em `C:\Windows\Java\Classes\classes.zip` (se o seu Windows estiver instalado no drive `C:\`) , portanto defina o `CLASSPATH` da seguinte forma:

```
set CLASSPATH=%CLASSPATH%;C:\Windows\Java\Classes\classes.zip
```

Se seu browser é Netscape 4, nas plataformas Windows95/98 o pacote está em no caminho {*diretório_de_instalação*}`\Program\Java\Classes\java40.jar`, e nos browsers Netscape 3, no caminho {*diretório_de_instalação*}`\Program\java\classes\java_30`, que devem ser usados para definir o `CLASSPATH` em cada caso.

Tendo-se definido o `CLASSPATH`, pode-se compilar, usando o compilador Java do JDK da Sun (ou outro compatível):

```
javac Desenha2.java
```

Corrija quaisquer erros e depois teste o applet, carregando-o no browser. Para ver possíveis mensagens de erro, abra o "Java Console" do seu browser. Após a correção de erros, e recompilação, pode ser necessário fechar todas as janelas do browser e abri-lo novamente para que a versão atualizada do applet seja carregada corretamente.

Conversão de tipos

Java é uma linguagem rigorosa em relação a tipos de dados. Na comunicação entre Java e JavaScript ocorrem diversas conversões de tipos. Quando valores são passados de Java para JavaScript, eles obedecem às conversões mostradas na tabela abaixo:

Tipo de dados Java	**Tipo de dados (objeto) JavaScript**
`byte`, `char`, `short`, `int`, `long`, `float`, `double`	*Number*
`boolean`	*Boolean*
`java.lang.String`	*String*

[3] O procedimento descrito é referente ao JDK da Sun, que roda em linha de comando.

`netscape.javascript.JSObject`	*Object*
`java.lang.Object`	*JavaObject*
`java.lang.Class`	*JavaClass*
tipo`[]`	*JavaArray*

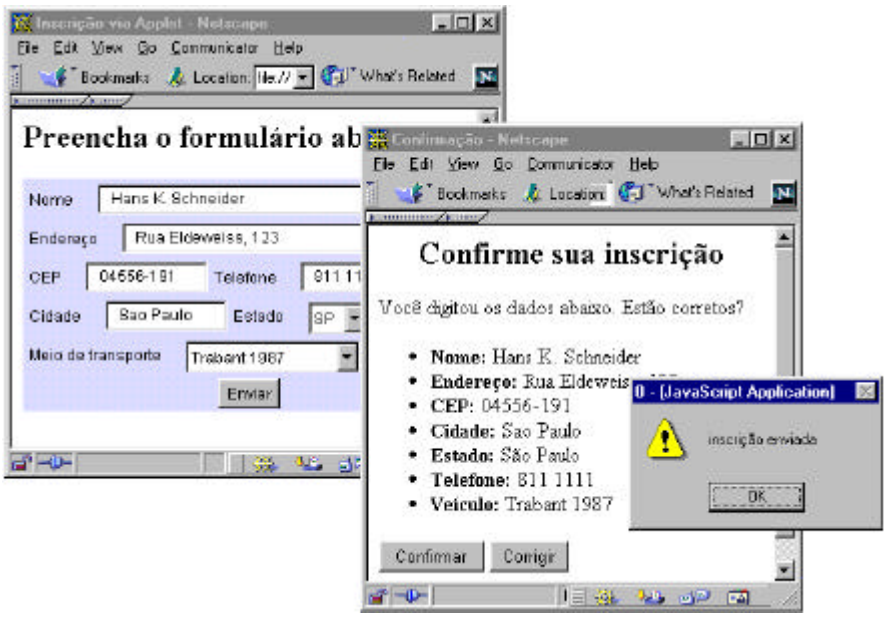
Os tipos de objeto *JavaObject* e *JavaArray* são representações JavaScript de objetos genéricos e vetores de qualquer tipo em Java. O tipo *JavaClass* refere-se à classes Java. É utilizado, por exemplo, para obter acesso a métodos estáticos.

Quando valores são passados de JavaScript para Java, a conversão obedece à tabela abaixo:

Tipo de dados (objeto) JavaScript	**Tipo de dados Java**
String	`java.lang.String`
Number	`java.lang.Float`
Boolean	`java.lang.Boolean`
JavaObject	`java.lang.Object`
Object	`netscape.javascript.JSObject`

Exercícios

12.2 Este exercício parece o exercício resolvido do capítulo 8. A aplicação, do ponto de vista do usuário, é idêntica. A diferença é que o formulário que deve ser preenchido desta vez é um applet! Ainda assim uma nova página HTML deve ser gerada em uma nova janela (como no capítulo 8). O que é necessário fazer aqui é realizar a comunicação Java-JavaScript para que o applet consiga criar a nova página. Use os arquivos `Formulario.html`, que contém a página HTML, e `Formulario.java`, código-fonte Java incompleto (falta implementar um método apenas).



12.3 Realize a validação dos campos do formulário do exercício anterior. Informe que os dados estão incorretos usando uma janela de alerta JavaScript (`alert()`).