ISEL

Instituto Superior de Engenharia de Lisboa

Departamento de Engenharia da Electrónica das Telecomunicações e dos Computadores

Infraestruturas Computacionais Distribuídas

WEB - Páginas Dinâmicas (client-side)
JavaScript - Applet's

Versão 1.2

Índice

Javascript – Introdução

- O *Javascript* é uma linguagem de *scripting* baseada em objectos que se insere numa página HTML dentro do elemento <script>
- Foi criada para ser facilmente embebida noutros produtos ou aplicações, como os Browsers
- O Browser interpreta todo o código *Javascript* que se encontrar dentro das etiquetas **<script>**...**</script>** **(Interpretado no lado do cliente)**
- *Foi inventada pela Netscape em 1996*
- A versão standard do *JavaScript* está a ser desenvolvida pelo ECMA (*European Computer Manufacturers Association*) e aprovada pelo ISO.
  - Documento da primeira versão oficial: ECMA-262 (ou ISO-16262).
  - O JavaScript 1.5 é completamente compatível com ECMA-262, edição 3

Javascript – Introdução (cont)

- Não tem nada haver com o java a não ser uma sintaxe semelhante
  - Grande parte da sintaxe dos mecanismos de controlo de fluxo do Java são iguais.
- Não é fortemente tipificada
- Links:
  - Core JavaScript Guide 1.5:
    - http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide
    - http://devedge.netscape.com/library/manuals/2000/javascript/1.5/guide/index.html (antigo link)
  - Referências do JavaScript
    - http://developer.mozilla.org/en/docs/JavaScript
    - http://devedge.netscape.com/central/javascript/ (antigo link)
  - Manual de referência incluindo tanto os objectos de *core* como os objectos das extensões para o lado do cliente e do servidor
    - http://www.jaws.umn.edu/javascript_1.1/contents.htm
  - Microsoft Jscript Manuals:
    - http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsoriJScript.asp

Utilização de JavaScript

- Dentro de documentos HTML

Vamos abordar o JavaScript de modo a realizar a validação de entrada de dados em formulários (antes de estes serem enviados para o servidor)

- Interpretado:
  - No lado do **cliente** (*client-side*)
    - Interpretado pelo browser
    - Páginas dinâmicas sem necessidade de interacção com o servidor
    - Utilização mais frequente
  - No lado do **servidor** (*server-side*)
    - Interpretado pelo servidor WEB
    - Páginas geradas dinamicamente na altura em que é recebido o pedido HTTP por parte do servidor (ex: SSI – Server Side Includes)
    - Utilização menos frequente

Javascript Vs Java (client-side)

JavaScript	Java
Interpretado no *browser* (não compilado)	O browser apenas recebe os *bytecodes* das *applets* e executa-os
Linguagem baseada em funções	Linguagem orientada a objectos
Podem ser adicionados a qualquer objecto propriedades e métodos dinamicamente	Não podem ser adicionadas dinamicamente às classes e às instancias propriedades e métodos
Os objectos não têm associado um tipo	As variáveis tem de ser declaradas de um tipo específico (fortemente tipificado)
O código está embebido nas páginas HTML	O código das *applets* está isolado das páginas HTML
As referências para objectos são verificadas em *run-time*	As referências são verificadas em *compile-time*

Javascript Vs Java (server-side)

- O Java executado do lado do servidor é muito mais potente que o *JavaScript*
- Java no servidor (*Servlet's/JSPs*)
  - Persistência
  - Lógica de negócios
  - Registo de acessos
  - Controle de acesso e segurança

Serviços que apenas fazem sentido do lado do servidor. Seja pela segurança ou pelo acesso a dados, necessitam de uma linguagem mais evoluída

Porquê JavaScript?

- Validar Formulários HTML antes de enviar os dados para serem tratados no servidor
  - Objectivo de reduzir o tempo de notificação do erro ao utilizador
- Interceptar e tratar eventos despoletados pelo utilizador durante a navegação
  - Objectivo de tornar as páginas reactivas alterando o seu conteúdo sem que tenham de ser geradas pelo servidor
- Interagir e controlar outras *frames* na mesma janela
- Para executar todo o tipo de *scripts* que façam sentido do lado do cliente

Etiquetas de Script

```
<script [type="mime-type"] [SRC="scriptURL"]>
<!– Código em JavaScript
    Esconder código Script de browsers que não o suportam
// –>
</script>
```

- mime-type:
  - text/javascript | text/vbscript
- scriptURL
  - Ficheiro com extensão .js que contém código JavaScript (para o caso do javaScript)

- *Script* na secção de cabeçalho (HEAD)
  - Código apenas interpretado caso seja chamado, ou caso ocorram eventos (declarações e definições)
- *Script* no corpo do documento (BODY)
  - O código é interpretado na altura em que a página é carregada

Exemplos de código JavaScript

Exemplo_def_externa.html

```
<html>
<head></head>
<body>
  <script type="text/javascript" SRC="xxx.js">
  </script>
</body>
</html>
```

xxx.js

```
Document.write("Este Script é externo")
```

Exemplo_body.html

```
<html>
<head></head>
<body>
<script type="text/javascript" >
<!–
    Document.write("HelloWorld!!!")
// –>
</script>
</body>
</html>
```

Exemplo_head.html

```
<html>
<head>
<script type="text/javascript" >
<!–
   function hello(){
     alert("HelloWorld!!!")
   }
// –>
</script>

</head>
<body onload="hello()">
</body>
</html>
```


Olá Mundo em *JavaScript*

simple.html

```
<!DOCTYPE HTML PUBLIC "-//W3C/DTD HTML
4.0//EN">
<html>
<head><title>A primeira p&aacute;gina de
JavaScript</title></head>
<body>
<h1>A minha primeira p&aacute;gina de
JavaScript</h1>

<script type="text/javascript">
<!–

document.write("<hr>");
document.write("<p>Ol&aacute;Mundo!!!</p>");
document.write("<hr>");

// –>
</script>

</body>
</html>
```

Sintaxe JavaScript – Tipos dinâmicos

- **As variáveis não têm tipo**
  - Apenas os valores têm tipo
  - A variável num determinado momento é do tipo do valor que contiver
- **Os tipos reconhecidos no JavaScript são os seguintes**:
  - Números (integer, float, etc…): 10, 42, 3.14159, etc
  - Valores lógicos (boolean): true ou false
  - Strings: "Olá Mundo!"
  - ***null***, palavra chave (***keyword***) especial que representa um valor nulo.
- **Existem duas formas de declarar variáveis:**
  - Com a palavra chave ***var.***
    - ***Ex: var x = 12; // int***
  - Sem a palavra chave ***var. Ex:***
    - ***x = 1.8; // float***
    - ***x = "Olá Pai Natal!!"; // String***

Sintaxe JavaScript – Declaração de funções

- A declaração de funções é feita utilizando a palavra chave ***function***
- O tipo de retorno não é declarado, nem os tipos dos argumentos
- Exemplos:

```
function dizOla(nome){
  return "Olá" + nome;
}
```

```
function factorial(num){
  if(num<=0)
    return 1;
  else
    return (num * factorial(num-1));
}
```


Sintaxe JavaScript – Classes e Objectos

- **Alteração dinâmica da estrutura de objectos**
  - Podem ser adicionados novas propriedades a objectos, bastando atribuir-lhes valores
  - Se o campo não existir, quando for feita a atribuição, então o JavaScript vai adicionar a propriedade
    - Ex:

```
var teste = new Object();
teste.campo1 = "Olá coelho da páscoa" //Criar o campo1
teste.campo2 = 55;                    //Criar o campo2
```

- **Pode ser utilizada notação literal da forma:**

  {campo1:val1, campo2:val2, …, campoN:valN}

```
var obj = new Object();
Obj.xpto = 3;
Obj.zeta = 7;
```

⇔

```
Obj2 = {xpto:3, zeta:7}
```


Sintaxe JavaScript – Classes e Objectos

- Acesso aos campos dos objectos:
  - A instrução "*for/in*" permite iterar as propriedades dos objectos:

```
for(nomeDoCampo in objecto){
  fazerAlgoComO(nomeDoCampo);
}
```

  - Podemos aceder directamente aos campos através de:
    - `objecto.nomeDoCampo`, ou
    - `objecto["nomeDoCampo"]`

Sintaxe JavaScript – Operadores

- O JavaScript define o seguinte conjunto de operadores:
  - Aritméticos:
    - `+, -, *, /, %`
  - Relacionais:
    - `> >= < <= == !=`
  - Lógicos:
    - `&&, ||`
  - Atribuição:
    - `=`
  - Atribuição composta: `+=, -=, *=, /=`

Exemplo de teste ao tipo de uma variável

- Exemplo de utilização da instrução de selecção múltipla **"SWITCH"**
- Para o teste do tipo é utilizado o **typeof**

```
switch ( typeof xpto ) {
  case 'number' : tipo = "número";
                  break;
  case 'string' : tipo = "String";
                  break;
  case 'boolean': tipo = "Boleano";
                  break;
  default:        tipo = "desconhecido";
}
```

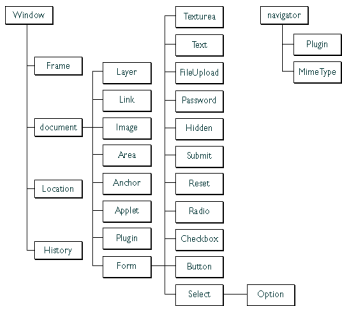

Objectos implícitos no Browser (*Client-Side*)

- Existem vários em JavaScript
- Representam várias entidades:
  - Página
  - Browser
  - Data
  - Matemática
  - String
- Deve-se usar a palavra 'javascript' antes do objecto implícito

```
<a href='javascript:history.back()'>Voltar</a>
```

Objectos implícitos no Browser (*Client-Side*)

- **Relação entre objectos relativos ao browser**:



- **Objectos *CORE do* JavaScript :**
  - **Array**
  - **Boolean**
  - **Date**
  - **Function**
  - **Math**
  - **Number**
  - **Object**
  - **String**

Aceder aos elementos HTML

- Exemplos de acesso aos elementos HTML:
- Utilizando o getElementById:
  - No *html* definir o elemento ***id*** das etiquetas:
    - `<p id="campoMsg"> </p>`
  - Aceder com o nome do elemento:
    - `document.getElementById("campoMsg").innerHTML ="Olá Mundo";`
  - Após a execução o elemento fica:
    - `<p id="campoMsg">Olá Mundo</p>`
- Utilizando o getElementsByName:
  - No *html* definir o elemento ***name*** das etiquetas:
    - `<p name="meuNomeElemento"> </p>`
  - Aceder com o nome do elemento:
    - `document.getElementsByName("meuNomeElemento")[0].innerHTML = "Olá Mundo";`

Objectos implícitos no Browser - Window

- Objecto `Window`
  - window.close() - Fecha a janela actual
  - window.open('param1', 'param2', 'param3') - Abre uma nova janela com os parâmetros seguintes:
    - param1: URL da página
    - Param2: título da página
    - param3: configurações da página

```
function abrirJanela(){
  window.open('http://www.deetc.isel.ipl.pt',
      'título',
      'menubar,scrollbars,resizable,height=600
      ,width=400');
}
```


Objectos implícitos no Browser - History

- Campos do Objecto `Window.History`
  - history.lenght - Número de endereços
  - history.back() - Voltar atrás
  - history.forward() - Avançar
  - history.go(n) - Ir para a página com o deslocamento (n)
- Exemplo de um link para voltar atrás:

```
<html><body>
…
<a href='javascript:history.back()'>Voltar</a>
</body></html>
```


Funções úteis para testes

- alert("Mensagem");
  - Abre uma janela de alerta com o texto passado como parâmetro.

```
<script type="text/javascript">
   alert("Hello World");
</script>
```

- document.write("Texto a ser inserido na página")
  - Usado para escrever texto no documento
  - Código HTML

```
<script type="text/javascript">
   document.write("<p>Ol&aacute; Mundo!</p>");
</script>
```


Tratamento de Eventos

- Dependem do Browser e da sua versão
  - Navigator – Netscape
  - Internet Explorer – Microsoft
- Eventos mais comuns:
  - onLoad, onUnload, onClick, onSubmit, onMouseOver, onMouseMove, onChange
  - Podem ser consultados em:
    - http://www.w3schools.com/tags/ref_eventattributes.asp
- Podem ser definidas funções para tratar a ocorrência de eventos (*Event Handlers*)
- Cada etiqueta lança determinados eventos que podem ser tratados por funções
  - Sintaxe para registar *Event Handlers*:

```
<etiqueta on[nomedoEvento]="nomeDaFuncao()"> … </etiqueta>
```

  - Ex:


```
<html>
<body onload="funcaoAbrir()">
...
</body></html>
```

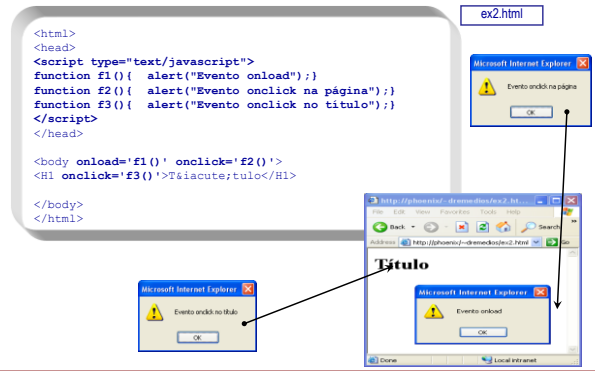
Eventos existentes

- **Listagem dos eventos recomendados pelo W3C:**
  - http://www.w3.org/TR/REC-html40/interact/scripts.html#h-18.2.3
- **BODY:**
  - onload=‘ ‘
  - onunload=‘ ‘
- **FORM:**
  - onreset=‘ ‘
  - onsubmit=‘ ‘
- **INPUT, SELECT, TEXTAREA:**
  - onblur=‘ ‘
  - onchange=‘ ‘
  - onfocus=‘ ‘
  - onselect=‘ ‘
- **BODY, P, FORM, TABLE, TR, TD, etc:**
  - **onclick=''**
  - **ondblclick=''**
  - **onkeydown=''**
  - **onkeypress=''**
  - **onkeyup=''**
  - **onmousedown=''**
  - **onmousemove=''**
  - **onmouseout=''**
  - **onmouseover=''**
  - **onmouseup=''**

Exemplo simples de eventos

ex2.html

```
<html>
<head>
<script type="text/javascript">
function f1(){  alert("Evento onload");}
function f2(){  alert("Evento onclick na página");}
function f3(){  alert("Evento onclick no título");}
</script>
</head>

<body onload='f1()' onclick='f2()'>
<H1 onclick='f3()'>T&iacute;tulo</H1>

</body>
</html>
```



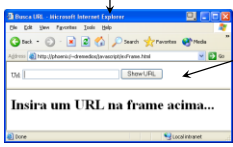
Exemplo de manipulação de Frames (1)

ex2.html

```
<html>
<head>
<title>Busca URL</title>
</head>
<frameset rows="150, *">
  <frame src="getURL.html" name="inputFrame">
  <frame src="DisplayURL.html" name="displayFrame">
</frameset>
</html>
```

getURL.html

```
<html>
<head>
<title>Busca URL</title>
<script type="text/javascript">
  function showURL(){
    var url = document.urlForm.urlField.value;
    parent.displayFrame.location = url;
    // ou parent.frames["displayFrame"].location = url;
  }

</script>
</head>
<body>
<form name="urlForm">
Url: <input type="text" name="urlField" size="32" />
<input type="button" value="Show URL" onclick="showURL()" />
</form>
</body>
</html>
```



Exemplo de manipulação de Frames (2)

Formulários HTML

(forma de recolher dados para enviar ao servidor)

Formulários HTML

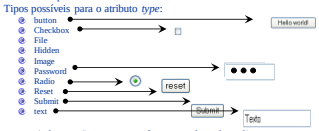
- É uma das opções disponíveis no HTML para fazer a recolha de dados de modo a serem enviados para o servidor;
- Permite especificar se queremos enviar os dados através do método HTTP **GET** ou **POST**;
  - **GET** – Os dados são concatenados no fim do URL no pedido **HTTP**.
  - **POST** – Os dados são inseridos no corpo da mensagem do pedido **HTTP**.

HTML – Formulários

- Elemento que permite a introdução de dados, pelo utilizador, no browser e envia para o servidor HTTP (ex: inserir palavras num motor de procura, responder a um inquérito)
- **<form>** … **</form>** – Delimita um formulário
  - **Action** – Atributo que indica o *URL para onde enviar os dados (CGI, Servlet, jsp, asp, …)*
  - **Method** – Atributo que especifica o método de envio dos dados
    - Method= "post" – Indica que é para ser enviada uma mensagem HTTP do tipo *POST com os dados no corpo da mensagem*
    - Method= "get" Indica que é para ser enviada uma mensagem HTTP do tipo *GET com os dados concatenados no URL*
  - **Target** – Indica a frame onde será visualizada a resposta
    - Pode ser uma frame especifica referindo o seu nome (ex: target="myframe1")
    - Pode ser uma frame relativa
      - Target="_blank" – Nova janela
      - Target="_self" – Na própria frame
      - Target="_parent" – No frameset pai
      - Target="_top" – No corpo da própria janela

HTML – Formulários (cont)

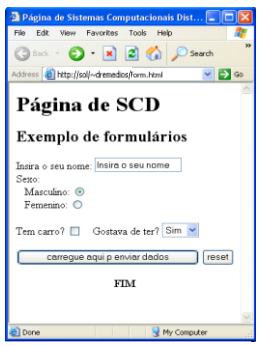
- Dentro de um formulário podemos ter os elementos:
  - <input> – Cria uma entrada no formulário.
    - Atributos – accept, align, alt, checked, disabled, maxlength, **name**, size, readonly, src, **type**, **value**
    - Não tem etiqueta de fim
    - Tipos possíveis para o atributo *type*:
      - button
      - Checkbox
      - File
      - Hidden
      - Image
      - Password
      - Radio
      - Reset
      - Submit
      - text



  - <select> … </select> – Permite especificar uma *drop down list*
    - *Os seus atributos são:*
      - *disabled*
      - *multiple*
      - *name*
      - *size*
    - <option> – introduz uma opção na lista, e os seu atributos são:
      - disabled
      - label
      - selected
      - value

  - <textarea> … </textarea> – Área em que pode ser introduzido texto (Ex: comentários)
    - name
    - Cols
    - Rows

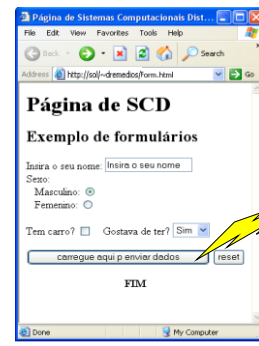
HTML – Formulários - Exemplo

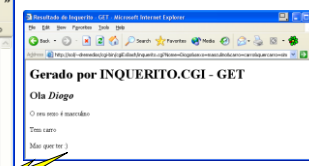


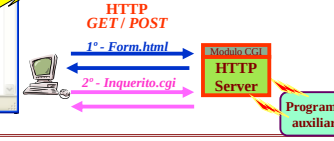
Envia os dados para o CGI inquerito.cgi através do método GET

```
<html>
<head>
 <meta http-equiv="Content-Type" content="text/html">
 <title>P&aacute;gina de Sistemas Computacionais Distribu&iacute;dos</title>
</head>

<body>
<h1>P&aacute;gina de SCD </h1>
<h2>Exemplo de formul&aacute;rios</h2>

<form method="GET" action="http://phoenix/~dremedios/cgi-bin/inquerito.cgi">
Insira o seu nome: <input type="text" size="17" name="None" value="Insira o seu
    nome">

<br/>
Sexo:
<br/>
   Masculino: <input type="radio" name="sexo" value="masculino"
    checked>
<br/>
   Femenino: <input type="radio" name="sexo" value="feminino">
<p></p>
Tem carro? <input type="checkbox" name="carro" value="carro">
   Gostava de ter?
<select name="quercarro">
 <option value="sim">Sim</option>
 <option value="não">Não</option>
</select>
<p></p>
<input type="submit" value="carregue aqui p enviar dados">
<input type="reset" value="reset">
</form>
<p style="text-align: center"><b>FIM</b></p>
</body>
</html>
```


HTML – Formulários - Exemplo

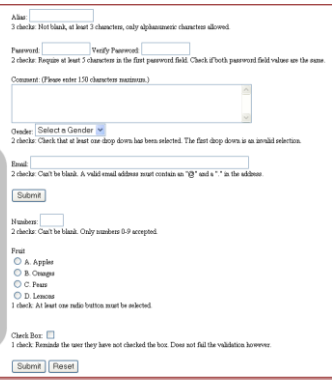






Exemplo de validação de Formulários

```
<html><head> ...
<script language="JavaScript">
   Function
Form1_Validator(theForm) {
   …
   }
</script>
</head>
<body>
...
 <form action="trataForm.jsp"
method="POST" onsubmit="return
Form1_Validator(this)"
name="Form1">
…
</form>
... </body><html>
```



Exemplo de validação de Formulários (1)

```
<html>
<head>
<script Language="JavaScript">
<!--
function Form1_Validator(theForm)
{

 var alertsay = ""; // define for long lines
 //alertsay is not necessary for your code,
 //but I need to break my lines in multiple lines
 //so the code won't extend off the edge of the page

/*ALIAS*/

 //check to see if the field is blank
 if (theForm.Alias.value == "")
 {
  alert("You must enter an alias.");
  theForm.Alias.focus();
  return (false);
 }

 //require at least 3 characters be entered
 if (theForm.Alias.value.length < 3)
 {
  alert("Please enter at least 3 characters in the \"Alias\" field.");
  theForm.Alias.focus();
  return (false);
 }

 //allow ONLY alphanumeric keys, no symbols or punctuation
 //this can be altered for any "checkOK" string you desire
 var checkOK =
       "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnop
       qrstuvwxyz0123456789";
 var checkStr = theForm.Alias.value;
 var allValid = true;
 for (i = 0;  i < checkStr.length;  i++)
 {
  ch = checkStr.charAt(i);
  for (j = 0;  j < checkOK.length;  j++)
  if (ch == checkOK.charAt(j))
  break;
  if (j == checkOK.length)
  {
   allValid = false;
   break;
  }
 }
 if (!allValid)
 {
  alert("Please enter only letter and numeric characters in the
       \"Alias\" field.");
  theForm.Alias.focus();
  return (false);
 }
```

Exemplo de validação de Formulários (2)

```
/*PASSWORD*/
 //require at least 5 characters in the password field
 if (theForm.Password.value.length < 5)
 {
  alert("Please enter at least 5 characters in the \"Password\" field.");
  theForm.Password.focus();
  return (false);
 }

 //check if both password fields are the same
 if (theForm.Password.value != theForm.Password2.value)
 {
  alert("The two passwords are not the same.");
  theForm.Password2.focus();
  return (false);
 }
/*COMMENT*/
 //allow only 150 characters maximum in the comment field
 if (theForm.comment.value.length > 150)
 {
  alert("Please enter at most 150 characters in the comment field.");
  theForm.comment.focus();
  return (false);
 }

/*SEX*/

 //check if no drop down has been selected
 if (theForm.sex.selectedIndex < 0)
 {
  alert("Please select one of the \"Gender\" options.");
  theForm.sex.focus();
  return (false);
 }
```

```
 //check if the first drop down is selected, if so, invalid selection
 if (theForm.sex.selectedIndex == 0)
 {
  alert("The first \"Gender\" option is not a valid selection.");
  theForm.sex.focus();
  return (false);
 }

/*EMAIL*/
 //check if email field is blank
 if (theForm.Email.value == "")
 {
  alert("Please enter a value for the \"Email\" field.");
  theForm.Email.focus();
  return (false);
 }
 //test if valid email address, must have @ and .
 var checkEmail = "@.";
 Var checkStr = theForm.Email.value;
 var EmailValid = false;
 var EmailAt = false;
 var EmailPeriod = false;
 for (i = 0;  i < checkStr.length;  i++)
 {
  ch = checkStr.charAt(i);
  for (j = 0;  j < checkEmail.length;  j++)
  {
```


Exemplo de validação de Formulários (3)

```
  if (ch == checkEmail.charAt(j) && ch == "@")
   EmailAt = true;
  if (ch == checkEmail.charAt(j) && ch == ".")
   EmailPeriod = true;
  if (EmailAt && EmailPeriod)
   break;
  if (j == checkEmail.length)
   break;
 }
 // if both the @ and . were in the string
 if (EmailAt && EmailPeriod)
 {
  EmailValid = true
  break;
 }
}
if (!EmailValid)
{
 alert("The \"email\" field must contain an \"@\" and a \".\".");
 theForm.Email.focus();
 return (false);
}

/*NUMBERS*/

 //check if numbers field is blank
 if (theForm.numbers.value == "")
 {
  alert("Please enter a value for the \"numbers\" field.");
  theForm.numbers.focus();
  return (false);
 }
```

```
 //only allow numbers to be entered
 var checkOK = "0123456789";
 var checkStr = theForm.numbers.value;
 var allValid = true;
 var allNum = "";
 for (i = 0;  i < checkStr.length;  i++)
 {
  ch = checkStr.charAt(i);
  for (j = 0;  j < checkOK.length;  j++)
   if (ch == checkOK.charAt(j))
    break;
  if (j == checkOK.length)
  {
   allValid = false;
   break;
  }
  if (ch != ",")
   allNum += ch;
 }
 if (!allValid)
 {
  alert("Please enter only digit characters in the \"numbers\" field.");
  theForm.numbers.focus();
  return (false);
 }
```


Exemplo de validação de Formulários (4)

```
 //require at least one radio button be selected
 var radioSelected = false;
 for (i = 0;  i < theForm.fruit.length;  i++)
 {
  if (theForm.fruit[i].checked)
   radioSelected = true;
 }
 if (!radioSelected)
 {
  alert("Please select one of the \"Fruit\" options.");
  return (false);
 }

/*CHECKBOX*/
 //alert if the box is NOT checked
 if (!theForm.checkbox1.checked)
 {
  alertsay = "Just reminding you that if you wish "
  alertsay = alertsay + "to have our Super Duper option, "
  alertsay = alertsay + "you must check the box!"
  alert(alertsay);
 }
 //because this is a sample page, don't allow to exit to the post action
 //comes in handy when you are testing the form validations and don't
 //wish to exit the page
 alertsay = "All Validations have succeeded. "
 alertsay = alertsay + "This is just a test page. There is no submission
          page."
 alert(alertsay);
 return (false);
 //replace the above with return(true); if you have a valid form to
          submit to
}
//--></script>
</head>
```

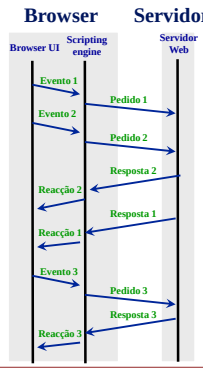
```
<body>
<form action="trataForm.jsp"
method="POST" onsubmit="return Form1_Validator(this)" name="Form1">
<input type="submit" name="Submit" value="Submit"><p>

Alias:
        <input type="text" size="15" maxlength="15" name="Alias"><br>
        3 checks:
        Not blank, at least 3 characters, only alphanumeric characters
        allowed.<p>

Password:
        <input type="password" size="10" maxlength="10" name="Password">

Verify Password:
        <input type="password" size="10" maxlength="10"
        name="Password2"><br>
        2 checks: Require at least 5 characters in the first password field.
        Check if both password field values are the same.<p>

Comment: (Please enter 150 characters maximum.)<br>
<textarea name="comment" rows="4" cols="50" wrap="virtual">
</textarea><br>

Gender:
        <select name="sex" size="1">
<option>Select a Gender</option>
<option value="M">Male</option>
```


Exemplo de validação de Formulários (5)

```
<option value="F">Female</option>
</select><br>
2 checks: Check that at least one drop down has been selected.
The first drop down is an invalid selection.<p>

Email:
        <input type="text" size="60" name="Email" value><br>
        2 checks: Can't be blank. A valid email address must contain an "@"
        and a "." in the address.<p>

<input type="submit" name="Submit" value="Submit"> <p>

Numbers: </strong>
        <input type="text" size="3" maxlength="3" name="numbers"><br>
        2 checks: Can't be blank. Only numbers 0-9 accepted.<p>

Fruit<br>
        <input type="radio" name="fruit" value="A"> A. Apples<br>
        <input type="radio" name="fruit" value="B"> B. Oranges<br>
        <input type="radio" name="fruit" value="C"> C. Pears<br>
        <input type="radio" name="fruit" value="D"> D. Lemons<br>
1 check: At least one radio button must be selected.<p>
<hr>

Check Box: <input type="checkbox" name="checkbox1" value="Y"><br>
1 check: Reminds the user they have not checked the box.
Does not fail the validation however.<p>
<p>

<input type="submit" name="Submit" value="Submit">
<input type="reset" name="Reset" value="Reset"><p>
</form>

</body>
```


AJAX

- Ajax - Asynchronous JavaScript And XML
- É uma técnica que permite criar interfaces mais ricas.
  - Quando queremos actualizar informação dentro da página, não implica o carregamento de toda a página.
- Com AJAX, o *javascript* pode comunicar directamente com o servidor, utilizando o objecto **XMLHttpRequest.** Com este objecto o nosso *javascript* pode trocar dados com o *servidor web*, sem ter de recarregar toda a página.
- Permite efectuar pedido HTTP assíncronos e registar funções em *javascript* (*handlers*) que irão tratar a resposta (dentro da mesma página).



AJAX

- AJAX utiliza a transferência assíncrona de dados (pedidos HTTP) entre o browser e o servidor web, permitindo que as páginas carreguem apenas pequenas quantidades de informação quando esta for necessária.
  - Exemplo 1: quando num formulário de uma página pretendemos pedir numa *option*: o país, cidade e localidade. Podemos carregar as cidades só depois de o utilizador fazer a escolha do país carregando apenas as cidades desse país.
  - Exemplo 2: Google suggest (2005)
- A técnica de AJAX faz com que a experiência de utilização das aplicações na internet seja mais rápida e mais *user-friendly*.
- É baseado nos standards:
  - **JavaScript**
  - **XML**
  - **HTML**
  - **CSS**

AJAX - Utilização

- Objecto **XMLHttpRequest**
  - Propriedade **onreadystatechange** – Permite registar a função que será invocada quando for recebida a resposta (ou em qualquer alteração do estado do pedido).

```
xmlHttp.onreadystatechange=function() {
   // tratar a resposta
}
```

  - Propriedade **readyState** – Permite consultar o estado do pedido.

```
State Description
0       The request is not initialized
1       The request has been set up
2       The request has been sent
3       The request is in process
4       The request is complete
```

  - Propriedade **responseText** – Permite aceder aos dados retornados na resposta.

AJAX - Utilização

- Efectuar um pedido
  - Open – Abre a ligação para o servidor
  - Send – Envia o pedido

```
xmlHttp.onreadystatechange=funcTrataResposta;
xmlHttp.open("GET",url,true);
xmlHttp.send(null);
```

- Recepção da resposta
  - Exemplo:

```
Function funcTrataResposta() {
  if (xmlHttp.readyState==4) {
       document.getElementById("txtHint").innerHTML
              =  xmlHttp.responseText;
  }
}
```


AJAX – Exemplo

- Sempre que detectada uma tecla premida ir buscar a hora actual ao servidor

testeDataAjax.html

```
<html> <head> <script src="utilsDate.js"></script> </head>
<body> <form name="myForm">
To get current Time <input type="text" value="Escreve Aqui!" onkeypress="GetDate();"/>
Time: <input id="time" type="text" name="time" /> </form> </body> </html>
```


AJAX - Exemplo

utilsDate.js

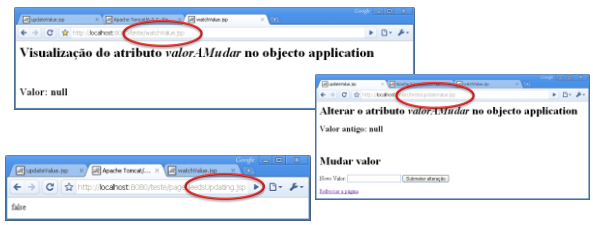
```
var xmlHttp;

function HandleReply() {
          if(xmlHttp.readyState == 4) {
            document.getElementById('time').value = xmlHttp.responseText;
          }
}
function GetDate() {
          xmlHttp = GetXmlHttpObject();
          xmlHttp.open("GET", "data.jsp", true);
          xmlHttp.onreadystatechange=HandleReply;
          xmlHttp.send(null);
}

function GetXmlHttpObject() {
  try { return new ActiveXObject("Msxml2.XMLHTTP");    } catch(e) {} // Internet Explorer
  try { return new ActiveXObject("Microsoft.XMLHTTP"); } catch(e) {} // Internet Explorer
  try { return new XMLHttpRequest();                   } catch(e) {} // Firefox, Opera 8.0+, Safari
  alert("XMLHttpRequest not supported");
  return null;
}
```

data.jsp

```
<%= new java.util.Date() %>
```

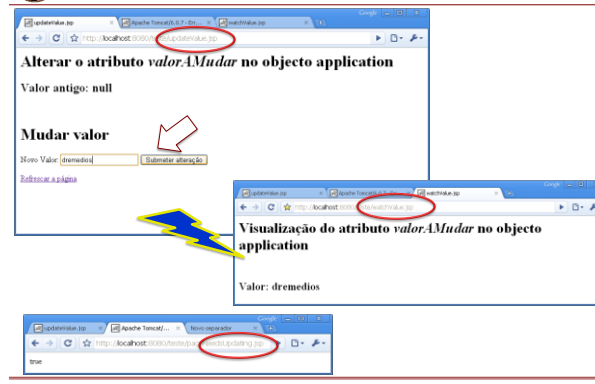

Auto refresh I

Exemplo de refrescamento automático quando o estado do servidor for alterado.

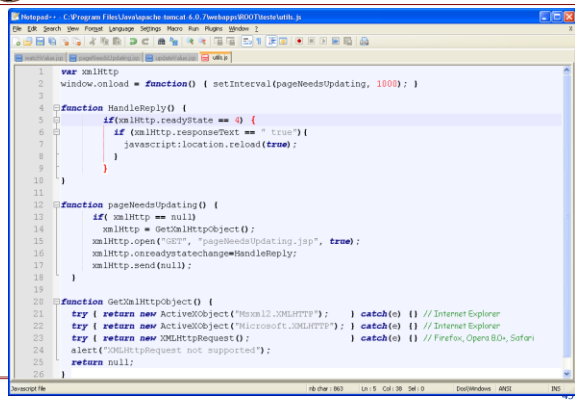


Útil para o trab. prático

Auto refresh II

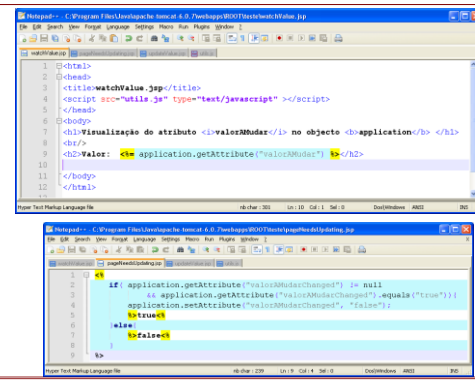


Auto refresh III



```
var XmlHttp
window.onload = function() { setInterval(pageNeedsUpdating, 1000); }

function HandleReply() {
        if(xmlHttp.readyState == 4) {
          if (xmlHttp.responseText == " true"){
            javascript:location.reload(true);
          }
        }
}

function pageNeedsUpdating() {
        if( xmlHttp == null)
          xmlHttp = GetXmlHttpObject();
        xmlHttp.open("GET", "pageNeedsUpdating.jsp", true);
        xmlHttp.onreadystatechange=HandleReply;
        xmlHttp.send(null);
    }

function GetXmlHttpObject() {
    try { return new ActiveXObject("Msxml2.XMLHTTP");    } catch(e) {} // Internet Explorer
    try { return new ActiveXObject("Microsoft.XMLHTTP"); } catch(e) {} // Internet Explorer
    try { return new XMLHttpRequest();                   } catch(e) {} // Firefox, Opera 8.0+, Safari
    alert("XMLHttpRequest not supported");
    return null;
}
```


Auto refresh IV

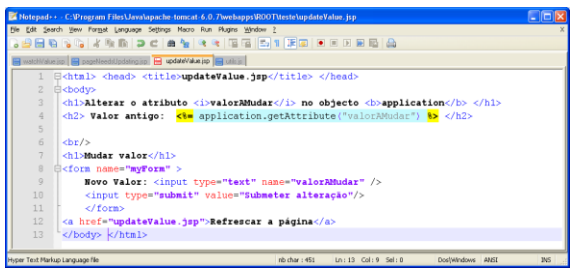


```
<html>
<head>
<title>watchValue.jsp</title>
<script src="utils.js" type="text/javascript" ></script>
</head>
<body>
<h1>Visualização do atributo <i>valorAMudar</i> no objecto <b>application</b> </h1>
<br/>
<h2>Valor:  <%= application.getAttribute("valorAMudar") %></h2>

</body>
</html>
```



```
<%
    if( application.getAttribute("valorAMudarChanged") != null
            && application.getAttribute("valorAMudarChanged").equals("true")){
        application.setAttribute("valorAMudarChanged", "false");
        %>true<%
    }else{
        %>false<%
    }
%>
```


Auto refresh V



```
<html> <head> <title>updateValue.jsp</title> </head>
<body>
<h1>Alterar o atributo <i>valorAMudar</i> no objecto <b>application</b> </h1>
<h2> Valor antigo:  <%= application.getAttribute("valorAMudar") %> </h2>

<br/>
<h1>Mudar valor</h1>
<form name="myForm" >
    Novo Valor: <input type="text" name="valorAMudar" />
    <input type="submit" value="Submeter alteração"/>
    </form>
<a href="updateValue.jsp">Refrescar a página</a>
</body> </html>
```


JavaScript - Conclusões

- O Javascript permite:
  - Que as páginas Web sejam mais dinâmicas
  - Adaptar as páginas baseadas na situação actual do cliente (ex: dimensão da janela)
  - Validar a introdução de dados nos formulários HTML
  - Controlar *Frames*
  - Interacção entre Java e JavaScript (*package LiveConnect*)
    - @ http://wp.netscape.com/eng/mozilla/3.0/handbook/javascript/livecon.htm
  - Mais informação sobre Javascript:
    - @ http://www.unix.com.ua/orelly/web/jscript/index.html
    - @ http://www.w3schools.com/js/
  - Mais informação sobre AJAX:
    - @ http://www.w3schools.com/Ajax/
    - @ http://www.tizag.com/ajaxTutorial/

Applet's

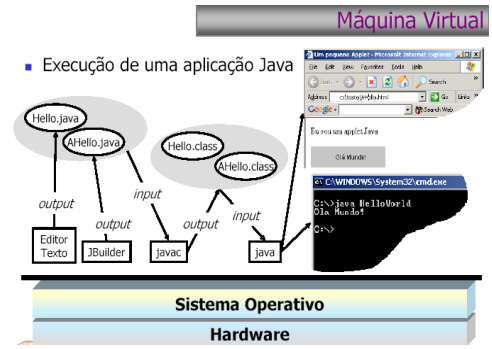
Introdução – Applets

- A linguagem de programação é **JAVA**
- É um pequeno programa cujo objectivo é ser embebido noutras aplicações.
- Um **applet** executa-se dentro do contexto do Browser (*client-side*) e apresenta uma aplicação gráfica em Java.
- Um **applet** executa-se na máquina do utilizador, e pode:
  - Desenhar imagens interactivamente, não só apresentar imagens estáticas
  - Responder directamente aos eventos do utilizador vindos do teclado e do rato
  - Executar cálculos na máquina do utilizador
- "Um **applet** transforma um rectângulo duma página web num motor computacional, completamente, interactivo tendo como suporte a JVM".

Introdução – Applets

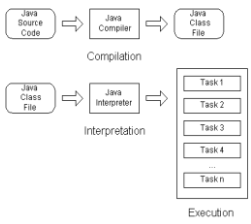
- É independente da plataforma (executa-se numa JVM).
- É um pedaço de código (.class) que é enviado do servidor (*web server*) para o cliente (*browser*) via HTTP.
- Permite aumentar a funcionalidade do browser de uma forma transparentemente para o utilizador.
- Há que ter em conta a segurança do código que vai ser executado.
- Introdução ao Java (Conceitos Básicos)
  - http://developer.java.sun.com/developer/onlineTraining/JavaIntro/contents.html
  - http://scv.bu.edu/Doc/Java/tutorial/index.html
  - http://www.falkhausen.de/en/diagram/diagram.html (Diagramas UML das classes)

Ambiente de Execução

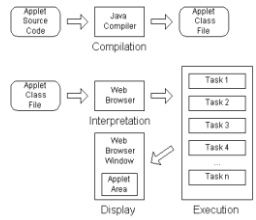


Aplicações Vs Applets

Aplicação Java



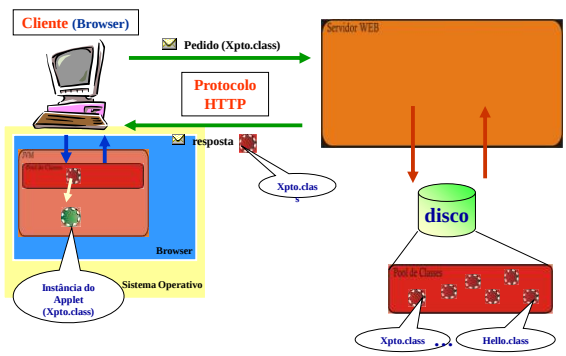
Applet Java



Visualização dos Applet's

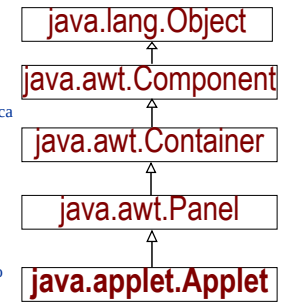
- Os **applets**, tipicamente, são visualizados, abrindo a página HTML, que lhe está associada, num *browser Java-enabled*.
- Os **applets** podem, também, ser visualizados com o utilitário "**appletviewer**" localizado na directoria ***bin*** da instalação do JDK
  - Recebe como primeiro parâmetro o ficheiro HTML, que utiliza a etiqueta `<Applet>` para embeber o Applet na página
    - Ex: `$appletviewer myAppletTest.html`

Ambiente de execução dos Applet's



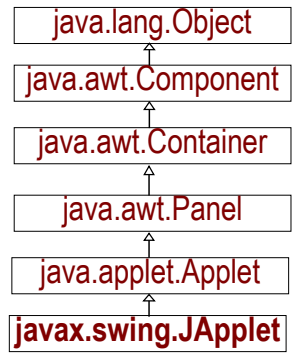
Criação de um Applet AWT

- **Criar uma SubClasse de java.applet.Applet**
  - Java.awt.Component
    - Objecto com representação gráfica e que interage com o utilizador
  - Java.awt.Container
    - Pode conter outros componentes
  - Java.awt.Panel
    - Contentor gráfico simples
  - Java.applet.Applet
    - Tem métodos que condicionam o seu ciclo de vida



Criação de um Applet SWING

- **Criar uma SubClasse de javax.swing.JApplet**
  - Javax.swing.JApplet
    - Uma versão extendida de Java.applet.Applet para dar suporte à arquitectura de componentes JFC/Swing
  - As principais diferenças são:
    - Poder conter componentes gráficos Swing;
    - Contém um **JRootPane** como único elemento raiz (**contentPane**).
      - Os componentes têm de o ter como contentor pai
      - Obtemo-lo com o método getContentPane()



Execução de um Applet

- O *browser* através do elemento `<applet>` e dos seus atributos sabe que tem de pedir, carregar e executar uma determinado *applet*
- Os atributos do elemento *applet* são:
  - Codebase: URL da directoria que contém o código (*.class)
  - Code: Nome da classe que vai executar (ex: hello.class)
  - Archive: Para incluir ficheiros JAR's
  - Name: Permite especificar um nome para o *applet*
  - Width e Height : Largura e altura da área reservada para o *applet*
  - Hspace e Vspace: Espaçamento à volta do *applet*
  - Alt: Texto alternativo para Browsers que não suportem *applets*
- O Browser carrega o *applet* através da rede para a máquina local
- Podem, opcionalmente, ser passados parâmetros para o *applet*
- É invocado o método INIT() do *applet*
- De seguida é invocado o método START()

Exemplo da invocação de um Applet

```
<html><head>
<title>HTML Test Page</title>
</head>
<body>
A Applet HelloWorld vai aparecer em
   baixo
<applet
  codebase = "."
  code     = "HelloWorldApplet.class"
  name     = "TestApplet"
  width    = "300"
  height   = "450"
  hspace   = "0"
  vspace   = "0"
  align    = "middle">

<param name="TEXT" value="HelloWorld"
  />
</applet>
</body></html>
```

Passar parâmetros para a *Applet* (neste caso o texto a mostrar)

Invocação de um *applet* utilizando o elemento *html* `<object>`

- A etiqueta ***Object*** é genérica para a inclusão de objectos de multimédia

```
<object
  classid = "clsid:CAFEEFAC-0014-0002-0000-ABCDEFFEDCBA"
  codebase = "."
  width = "300" height = "60" >
    <param name = "CODE" value = "HelloWorldApplet.class" >
    <param name = "type" value = "application/x-java-applet;jpi-
       version=1.4.2">
    <param name = "scriptable" value = "false">
    <param name = "myData" value = "Olá Mundo enviado da página HTML
       (Object)">

  ERRO: Na inserção da Applet com o elemento OBJECT
</OBJECT>
```

- Mais informações sobre o elemento ***"OBJECT"*** em:
  http://www.w3.org/TR/REC-html40/struct/objects.html

Invocação de um *applet* utilizando o elemento *html* `<object>`

- **`classid`**
  - Atributo que identifica a versão do Java Plug-in que o browser deve utilizar
  - A classid seguinte indica ao browser para utilizar o Plug-in mais recente que estiver instalado
    - classid="clsid:8AD9C840-044E-11D1-B3E9-00805F499D93"
  - Em alternativa pode-se indicar uma versão específica utilizando:
    - classid="clsid:CAFEEFAC-xxxx-yyyy-zzzz-ABCDEFFEDCBA"
    - Onde "xxxx", "yyyy", and "zzzz" são números de 4 dígitos que identificam uma versão especifica
    - Exemplo, para Java Plug-in versão 1.6.0:
      - classid="clsid:CAFEEFAC-0016-0000-0000-ABCDEFFEDCBA"
- Só funciona no Internet Explorer

Invocação de um *applet* utilizando o elemento *html* `<embed>`

- Apenas funciona com a família de *browsers* Mozilla

```
<embed
       code="Applet1.class",
       width="200",
       height="200",
       type="application/x-java-applet;version=1.6.0"
       pluginspage="http://java.sun.com/javase/downloads"/>
```

- Para mais informações consultar:
  http://java.sun.com/docs/books/tutorial/deployment/applet/mixedbrowser.html

Ciclo de Vida de um Applet

- **Os *applets* não têm a função `main()` como as aplicações normais.**
- **Em vez disso estão definidos os seguintes métodos para gerir o ciclo de vida de um *applet*:**
  - **Init()**
    - Invocado após o *applet* ter sido instanciado para o sistema por parte do *browser* ou *applet viewer* com o objectivo de realizar as iniciações.
  - **Start()**
    - Invocado quando a página é carregada
    - E de cada vez que a página é activada
  - **Stop()**
    - Invocado quando o *applet* deixa de ser visualizado, seja pela janela ter sido minimizada ou por ter sido fechada.
  - **Destroy()**
    - Invocado para informar que o *applet* vai ser destruído e que este deve libertar todos os recursos que tiver atribuídos.
- **Estes métodos são invocados pelo *browser* e, tipicamente, não são explicitamente chamados**

Ciclo de Vida de um Applet

- O objectivo é carregar o *applet* apenas no primeiro acesso chamando o `init()`, após estar carregado seriam chamados o `start()` e o `stop()` quando a janela fosse visualizada ou deixasse de o ser, sendo apenas chamado o `destroy()` quando a instância do *applet* fosse destruída.
- Existem browsers que contornam ligeiramente estas regras, nomeadamente o IE.
  - Quando se entra na página com o *Applet* são invocados sempre o seu `init()` e o `start()` e quando se sai o `stop()` e o `destroy()`.
- Protótipo para um *applet* simples:

```
Class MySimpleApplet extends java.applet.Applet {
    void init()     {}
    void start()    {}
    void stop()     {}
    void destroy()  {}
}
```


Outros métodos definidos nos Applet's

- **Obter o parâmetro referido na *string* que foi passado no HTML**
  - **String getParameter(String)**
- **Método que é chamado para desenhar o *Applet***
  - **paint(Graphics)**
- **Obter o URL base do código do *Applet***
  - **String getCodeBase()**
- **Pedir para mostrar uma dada *string* na "status window"**
  - **showStatus(String)**

Exemplo – HelloWorld.java

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;

public class HelloWorldApplet extends Applet {

  Label myHelloLabel = new Label();

  //Initialize the applet
  public void init() {
   try {
      myHelloLabel.setFont(new java.awt.Font("Dialog", 0, 35));
      myHelloLabel.setText(getParameter("TEXT", "Default Text!!"));
      myHelloLabel.setBounds(new Rectangle(112, 95, 199, 39));
      this.setLayout(null);
      this.add(helloLabel1, null);
    }catch(Exception e) {
      e.printStackTrace();
    }
  }

  //Start the applet
  public void start() {
  }
  //Stop the applet
  public void stop() {
  }
  //Destroy the applet
  public void destroy() {
  }

}
```

Descobrir o parâmetro "TEXT", caso este tenha sido fornecido. Caso contrário será utilizada a string "Default Text!"

Restrições de segurança

- **Os *applets* como aplicações externas, a correrem nos computadores dos utilizadores, têm as seguintes restrições:**
  - Verificação dos *Byte-codes* de modo a garantir que estes não foram corrompidos na rede
  - Acesso ao sistema limitado – Não podem lançar processos na máquina local
  - Não têm acesso ao sistema de ficheiros
  - Operações de rede limitadas
    - Apenas podem efectuar ligações por *sockets* para o *host* de onde foi feito o *download* do *Applet* (.class).
- **Estas restrições podem ser levantadas caso o applet esteja assinado digitalmente**

Restrições de segurança

- A segurança nos *applets* é fornecida pelo browser onde o applet é carregado
- As políticas de segurança de um browser são configuráveis
- Os browser podem impor restrições mais leves aos *applets* que tenham sido assinados digitalmente
- Um *applet* assinado verifica a origem do *applet*, mas não garante nada sobre o seu comportamento

Ficheiros Java ARchive (JAR)

- Criação de um ficheiro JAR
  - Exemplo da linha de comandos:

```
@ $ jar cvf SomeJarName.jar SomeApplet.class
  HelperClass.class picture.gif
```

Lista de ficheiros a incluir

- Carregar um *applet* referenciando um JAR
  - Etiqueta HTML de exemplo:

```
@ <APPLET CODE="SomeApplet.class"
          ARCHIVE="SomeJarName.jar"
          WIDTH=400 HEIGHT=300>
  </APPLET>
```


Como assinar um Applet

- O *applet* tem de estar empacotado num ficheiro **JAR**
- É necessário gerar um certificado
  - `$ keytool -genkey -alias scd -keypass scdpass -keystore scdchaves`
  - `$ keytool -export -alias scd -keystore scdchaves -file scd.cer`
- Assinar o Java ARchive
  - `$ jarsigner -keystore scdchaves -storepass scdpass -keypass scdpass SomeJarName.jar scd`
- Mais informação sobre a aplicação ***"keytool"***

  http://java.sun.com/j2se/1.3/docs/tooldocs/win32/keytool.html