ACTIONSCRIPT

AMANDA ALVES | ISABELLY ARAUJO | LARA MACÊDO

ACTIONSCRIPT

[2006]

- CRIADA POR GARY GROSSMAN (1998).
- BASEADA EM ECMASCRIPT, ASSIM COMO AS LINGUAGENS JAVASCRIPT E JAVA.
- COMPILADA EM ADOBE FLASH; ADOBE FLEX.
- MULTIPARADIGMA
- TIPAGEM FORTE, ESTÁTICA E ROBUSTA

APLICABILIDADE

DESENVOLVIMENTO DE APLICAÇÕES RIA (RICH INTERNET APPLICATIONS)

AUTODESK®
PIXLR®

INSTALAÇÃO E USO

ACTIONSCRIPT

INSTALAÇÃO

- **PASSOS**

COMO SABEMOS, ACTIONSCRIPT É A LINGUAGEM DE CRIAÇÃO DE SCRIPTS DO ADOBE FLASH, QUE JÁ VEM INSTALADO COM ELA. PARA EXECUTAR A INSTALAÇÃO, SIGA AS INSTRUÇÕES A SEGUIR:

- VOCÊ PRECISA TER O ARQUIVO **.EXE** DO ADOBE FLASH BAIXADO EM SEU COMPUTADOR PARA PODER INICIAR A INSTALAÇÃO;

- EXECUTE O INSTALADOR. EM SEGUIDA, CLIQUE EM **"SIM"** . CASO SEJA NECESSÁRIO AUTORIZAR A INSTALAÇÃO, INSIRA O USUÁRIO E A SENHA DE UM ADMINISTRADOR;

INSTALAÇÃO

- **PASSOS**
- ESCOLHA A FORMA QUE DESEJA RECEBER AS ATUALIZAÇÕES DO PROGRAMA E, EM SEGUIDA, CLIQUE EM **“PRÓXIMO”**;
- AGUARDE A CONCLUSÃO DO DOWNLOAD E A INSTALAÇÃO;
- CLIQUE EM **“CONCLUIR”**.

USO

- PASSO A PASSO
- ABRA O FLASH, CLIQUE EM “FILE > NEW”, SELECIONE O ARQUIVO ACTIONSCRIPT E CLIQUE EM “OK”. UMA CAIXA DE DIÁLOGO IRÁ SE ABRIR, AGORA SIGA OS PASSOS ABAIXO PARA COMPILAR O SEU CÓDIGO.

- COMPILANDO...

USO

1. ESCREVA O CÓDIGO E SALVE O ARQUIVO.

EXEMPLO:

```
package {
 import flash.display.Sprite;
 import flash.text.TextField;

 public class Hello extends Sprite{

  [SWF(width="500", height="500")]
  public function Hello () {
      var hello:TextField = new TextField();
      hello.text = "Hello world!";
      hello.x = 100;
      hello.y = 100;
      addChild(hello);
  }
}
```

USO

2. ABRA O PROMPT DE COMANDO. VÁ ATÉ A PASTA ONDE VOCÊ SALVOU O ARQUIVO ACIMA E DIGITE:

```
$ mxmlc nomedoarquivo.as
```

3. APÓS A COMPILAÇÃO, CONFIRA SE EXISTE UM ARQUIVO .SWF COM O MESMO NOME DO ARQUIVO .AS QUE JÁ FOI SALVO.

4. CLIQUE EM ARQUIVO, DEPOIS ABRIR E SELECIONE O ARQUIVO .SWF DESEJADO.

SINTAXE BÁSICA

ACTIONSCRIPT

VARIÁVEIS E CONSTANTES

VARIÁVEIS

- STRING
- NUMBER
- INT
- UINTDATE
- BOOLEAN
- OBJECT
- CLASS
- ARRAY

DECLARANDO:

- VARIÁVEL

```
var myPhoneNumper: Number;
```

- CONSTANTE

```
const taxa_desconto:Number = 0.07;
```

OPERADORES RELACIONAIS, LÓGICOS E ARITMÉTICOS

RELACIONAIS

- >
- <
- =>
- <=
- ==
- !=

ARITMÉTICOS

- +
- -
- /
- *
- %

LÓGICOS

- &&
- ||

ESTRUTURAS DE CONTROLE CONDICIONAL

SWITCH

```
switch (myVariable)
{
        case 1:
        statements;
        break;
}
```

IF

```
if(condition){
        statements;
}
```

ESTRUTURAS DE CONTROLE CONDICIONAL

ELSE IF /ELSE

```
var idade:Number = 20;
if(idade > 18){
        trace("Bem-Vindo(a)!");
}
else if(idade < 18){
        trace("Ops, você não deve estar aqui");
}
else
        trace("Forbidden");
```

ESTRUTURAS DE REPETIÇÃO

WHILE

```
var i:Number = 0;
while(i < 10){
        trace(i);
        i++;
}
```

DO WHILE

```
var i:Number = 0;
do{
        trace(i);
        i++;
} while (i < 10);
```

ESTRUTURAS DE REPETIÇÃO

FOR

```
for(var i:Number=0; i<10; i++){
        trace(i);
}
```

FOR EACH

```
var obj:Object = new objeto();
obj.i = "Olá";
obj.n = 12;
foreach(var f in obj)
{
        trace(f);
}
```

ESTRUTURAS DE REPETIÇÃO

FOR IN

```
var obj.Object = new objeto();
obj.i = "Olá!";
obj.n = 12;
for (var f in obj){
        trace(f);
}
```

VETORES, MATRIZES E STRINGS

- STRING
  - SUBSTRING

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.substr(11,15)); //output: Paris, Texas!!!
trace(str.substring(11,15)); //output: Pari
```

```
var str:String = "Hello from Paris, Texas!!!";
trace(str.slice(11,15)); // output: Pari
trace(str.slice(-3,-1)); // output: !!
trace(str.slice(-3,26)); // output: !!!
trace(str.slice(-3,str.length)); // output: !!!
trace(str.slice(-8,-3)); // output: Texas
```

VETORES, MATRIZES E STRINGS

- STRING
  - —UPPER/LOWER CASE

```
var str:String = "Dr. Bob Roberts, #9."
trace(str.toLowerCase()); //dr. bob roberts, #9.
trace(str.toUpperCase()); //DR. BOB ROBERTS, #9.
```

VETORES, MATRIZES E STRINGS

- STRING
  - SORT

```
var str:String = "hello world!";
for (var i:int = 0; i < str.lengh; i++)
{
        trace(str.charAt(i), "-", str.charCodeAt(i));
}
```

VETORES, MATRIZES E STRINGS

- STRING
  - —CONCATENAÇÃO

```
var str1:String = "green";
var str2:String = "ish";
var str3:String = str1 + str2; // str3 == "greenish"
```

```
var str:String = "green";
str += "ish"; // str == "greenish"
Além disso, a classe String inclui um método concat() que pode ser usado da seguinte maneira:
var str1:String = "Bonjour";
var str2:String = "from";
var str3:String = "Paris";
var str4:String = str1.concat(" ", str2, " ", str3);
// str4 == "Bonjour from Paris"
```

VETORES, MATRIZES E STRINGS

- ARRAY
  - VETOR

```
var myArray:Array = ["Flash", "ActionScript"];
```

VETORES, MATRIZES E STRINGS

- ARRAY
  - —VETOR: PROPRIEDADES

```
var poets:Array = ["Blake", "cummings", "Angelou", "Dante"];
poets.sort(); // default sort
trace(poets); // output: Angelou,Blake,Dante,cummings

poets.sort(Array.CASEINSENSITIVE);
trace(poets); // output: Angelou,Blake,cummings,Dante

poets.sort(Array.DESCENDING);
trace(poets); // output: cummings,Dante,Blake,Angelou

poets.sort(Array.DESCENDING | Array.CASEINSENSITIVE); // use two options
trace(poets); // output: Dante,cummings,Blake,Angelou
```

VETORES, MATRIZES E STRINGS

- ARRAY
  - VETOR: PROPRIEDADES

```
var myArray:Array = ["Flash", "ActionScript", "Republic of Code"];
myArray[3] = "Tutorial";
trace(myArray);
```

```
var myArray:Array = ["Flash", "ActionScript", "Republic of Code"];
myArray.push("Tutorials");
trace(myArray);
```

```
var myArray:Array = ["Flash", "ActionScript", "Republic of Code"];
myArray.splice(2,1);
trace(myArray);
```

```
var myArray:Array = ["Flash", "ActionScript", "Republic of Code"];
myArray.pop();
trace(myArray);
```

VETORES, MATRIZES E STRINGS

- ARRAY
  - MATRIZ MULTIDIMENSIONAL: DUAS MATRIZES INDEXADAS

```
var ListaTarefas:Array = new Array();
ListaTarefas[0] = ["wash dishes", "take out trash"];
ListaTarefas[1] = ["wash dishes", "pay bills"];
ListaTarefas[2] = ["wash dishes", "dentist", "wash dog"];
ListaTarefas[3] = ["wash dishes"];
ListaTarefas[4] = ["wash dishes", "clean house"];
ListaTarefas[5] = ["wash dishes", "wash car", "pay rent"];
ListaTarefas[6] = ["mow lawn", "fix chair"];
```

```
traceListaTarefas[2][1]); // output: dentist
```

VETORES, MATRIZES E STRINGS

- ARRAY
  - —MATRIZ ASSOCIATIVA COM UMA MATRIZ INDEXADA

```
var ListaTarefas:Object = new Object();
ListaTarefas["Monday"] = ["wash dishes", "take out trash"];
ListaTarefas["Tuesday"] = ["wash dishes", "pay bills"];
ListaTarefas["Wednesday"] = ["wash dishes", "dentist", "wash dog"];
ListaTarefas["Thursday"] = ["wash dishes"];
ListaTarefas["Friday"] = ["wash dishes", "clean house"];
ListaTarefas["Saturday"] = ["wash dishes", "wash car", "pay rent"];
ListaTarefas["Sunday"] = ["mow lawn", "fix chair"];
```

FUNÇÕES

- FUNÇÕES BÁSICAS

```
function Nome_Funcao(argumento):returnType{
        statements;
}
```

SINTAXE OO

ACTIONSCRIPT

CLASSES

- SINTAXE PARA DEFINIÇÃO DE UMA CLASSE:

```
public class nome_da_classe_sem_espacos{
//corpo da classe
}
```

CLASSES

- EXEMPLO

```
public class carro
{
public var placa:String = “MM 0056”;
private var volume_tanque:Number = 30;

public function acender_farol(){
return 1;
}

private var queimar_gasolina(){
return 1;
}
static function getNumeroRodas(){
return 4;
}
}
```

OBJETOS

- SINTAXE PARA A CRIAÇÃO (INSTANCIAÇÃO) DE UM OBJETO

```
var nomeDoObjeto: nomeDaClasse= new nomeDaClasse();
```

MÉTODOS E ATRIBUTOS (VISIBILIDADE)

Tipo	Definição
internal (padrão)	Visível para referências dentro do mesmo pacote.
private	Visível para referências na mesma classe.
protected	Visível para referências na mesma classe e em classes derivadas.
public	Visível para referências em todos os lugares.
static	Especifica que uma propriedade pertence à classe, ao contrário das ocorrências da classe.
UserDefinedNamespace	Nome do espaço para nomes personalizado definido pelo usuário.

CONSTRUTORES

- O CÓDIGO DE MÉTODO DE CONSTRUTOR É EXECUTADOTODA VEZ QUE UMA OCORRÊNCIA DA CLASSE É CRIADA COM *NEW.*

EXEMPLO:

```
class Example
{
 public var status:String;
 public function Example()
 {
 status = "initialized";
 }
}
var myExample:Example = new Example();
trace(myExample.status); // output: initialized
```

OBSERVAÇÃO: CONSTRUTORES SÓ PODEM SER PÚBLICOS, SENDO ASSIM OPCIONAL A UTILIZAÇÃO DO ATRIBUTO *PUBLIC.*

HENRANÇA

- É UTILIZADA A PALAVRA *EXTENDS* PARA PARA INDICAR QUE UMA CLASSE HERDA DE OUTRA CLASSE.

- SE UMA PROPRIEDADE FOR DECLARADA COMO PÚBLICA, ELA SERÁ VISÍVEL EM QUALQUER LUGAR DE CÓDIGO. AO CONTRÁRIO DAS PROPRIEDADES DECLARADAS COMO: *PRIVATE, PROTECTED E INTERNAL*, A *PUBLIC* NÃO COLOCA NENHUMA RESTRIÇÃO SOBRE A HERANÇA DA PROPRIEDADE.

HERANÇA E POLIMORFISMO

- NA HERANÇA, PODE-SE USAR O POLIMORFISMO DO CÓDIGO, COMO NO EXEMPLO DO PRÓXIMO SLIDE, ONDE, DEVIDO AO POLIMORFISMO A CLASSE *AREA()* VAI FAZER OS RESPECTIVOS CÁLCULOS PARA OS OBJETOS DO TIPO *SQUARE* E *CIRCLE*:

HERANÇA E POLIMORFISMO

```
class Shape
{
    public function area():Number
    {
        return NaN;
    }
}

class Circle extends Shape
{
    private var radius:Number = 1;
    override public function area():Number
    {
        return (Math.PI * (radius * radius));
    }
}

class Square extends Shape
{
    private var side:Number = 1;
    override public function area():Number
    {
        return (side * side);
    }
}

var cir:Circle = new Circle();
trace(cir.area()); // output: 3.141592653589793
var sq:Square = new Square();
trace(sq.area()); // output: 1
```

SOBRECARGA

- A SOBRECARGA (*OVERLOAD*), MÉTODOS DE MESMO NOME, QUE TEM COM ARGUMENTOS DIFERENTES (SEJA EM NÚMERO E/OU TIPO DE DADO).

- O ACTIONSCRIPT NÃO SUPORTA SOBRECARGA.

EXCEÇÕES

- O AS3 DÁ SUPORTE AO TRATAMENTO DE EXCESSÕES ATRAVÉS DA CLÁUSULA *TRY/CATCH/FINALLY* E DAS CLASSES DE ERRO.

EXCEÇÕES (TIPOS)

CATEGORIAS DE EXCEÇÕES:

- ERROS DE TEMPO DE COMPILAÇÃO: OCORREM QUANDO PROBLEMAS SINTÁTICOS NO SEU CÓDIGO IMPEDEM A CRIAÇÃO DO APLICATIVO.
- ERROS DE TEMPO DE EXECUÇÃO: OCORREM QUANDO VOCÊ EXECUTA O APLICATIVO DEPOIS DE COMPILÁ-LO.
- ERROS SÍNCRONOS: SÃO ERROS DE TEMPO DE EXECUÇÃO QUE OCORREM NO MOMENTO EM QUE UMA FUNÇÃO É CHAMADA.
- ERROS ASSÍNCRONOS: SÃO ERROS DE TEMPO DE EXECUÇÃO QUE OCORREM EM VÁRIOS MOMENTOS DURANTE O TEMPO DE EXECUÇÃO.
- EXCEÇÕES NÃO DETECTADAS: SÃO ERROS LANÇADOS SEM NENHUMA LÓGICA CORRESPONDENTE (COMO UMA INSTRUÇÃO CATCH) EM RESPOSTA A ELAS.

CAPTURA E LANÇAMENTO DE EXCEÇÕES

- DEFINE-SE UM BLOCO *TRY* PARA VERIFICAR SE CÓDIGO POSSUI ALGUM ERRO, SE ESSE FOR O CASO É EXECUTADO O BLOCO CONTIDO EM *CATCH*.

- A INSTRUÇÃO *FINALLY* DELIMITARÁ AS INSTRUÇÕES QUE SERÃO EXECUTADAS CASO OCORRA OU NÃO UM ERRO NO BLOCO *TRY*. SE NÃO HOUVER NENHUM ERRO, AS INSTRUÇÕES NO BLOCO *FINALLY* SERÃO EXECUTADAS DEPOIS QUE AS INSTRUÇÕES DO BLOCO *TRY* FOREM CONCLUÍDAS. OS BLOCOS *CATCH* E *FINALLY* SÃO OPCIONAIS, PORÉM, É PRECISO QUE UM DELES ESTEJA PRESENTE; CASO CONTRÁRIO SERÁ LANÇADO UM ERRO DE COMPILAÇÃO.

CAPTURA E LANÇAMENTO DE EXCEÇÕES

```
var MyError:Error = new Error("Encountered an error with the numUsers value", 99);
var numUsers:uint = 0;
try
{
    if (numUsers == 0)
    {
        trace("numUsers equals 0");
    }
}
catch (error:uint)
{
    throw MyError; // Catch unsigned integer errors.
}
catch (error:int)
{
    throw MyError; // Catch integer errors.
}
catch (error:Number)
{
    throw MyError; // Catch number errors.
}
catch (error:*)
{
    throw MyError; // Catch any other error.
}
finally
{
    myFunction(); // Perform any necessary cleanup here.
}
```

CRIAÇÃO DE NOVAS EXCEÇÕES

- COM O *THROW*, VOCÊ PODE LANÇAR ERROS EXPLICITAMENTE.
- TAMBÉM PODE-SE ESTENDER UMA DAS CLASSES *ERROR* PADRÃO PARA CRIAR SUAS PRÓPRIAS CLASSES DE ERRO NO ACTIONSCRIPT, VOCÊ PODE USÁ-LAS PARA IDENTIFICAR ERROS OU GRUPOS DE ERROS ESPECÍFICOS QUE SÃO EXCLUSIVOS DO SEU APLICATIVO OU PARA FORNECER RECURSOS DE EXIBIÇÃO DE ERROS EXCLUSIVOS PARA OS ERROS GERADOS PELO SEU APLICATIVO.
- UMA CLASSE DE ERRO ESPECIALIZADA DEVE ESTENDER A CLASSE *ERROR* PRINCIPAL DO ACTIONSCRIPT. VEJA UM EXEMPLO DE UMA CLASSE *APPERROR* ESPECIALIZADA QUE ESTENDE A CLASSE *ERROR*.

CLASSE ERROR

- A CLASSE *ERROR*:

```
public class AppError extends Error
{
    public function AppError(message:String, errorID:int)
    {
        super(message, errorID);
    }
}
```

- *APPERROR* NO PROJETO:

```
try
{
    throw new AppError("Encountered Custom AppError", 29);
}
catch (error:AppError)
{
    trace(error.errorID + ": " + error.message)
}
```

SINTAXE FUNCIONAL

ACTIONSCRIPT

CONCEITOS DE FUNÇÕES BÁSICAS

- CHAMADA DE FUNÇÕES
- QUANDO UTILIZAMOS O IDENTIFICADOR DE UMA FUNÇÃO SEGUIDO DE PARÊNTESES, UMA FUNÇÃO É CHAMADA.
  - EXEMPLO:

```
trace()
```

CONCEITOS DE FUNÇÕES BÁSICAS

- CHAMADA DE FUNÇÕES
- SE UMA FUNÇÃO ESTIVER COM OS PARÊNTESES VAZIOS, SIGNIFICA QUE ELA NÃO POSSUI UM PARÂMETRO.
  - EXEMPLO:

```
var randomNum:Number = math.random()
```

CONCEITOS DE FUNÇÕES BÁSICAS

- INSTRUÇÕES DE FUNÇÃO
  - PARA DEFINIR UMA FUNÇÃO, É NECESSÁRIO SEGUIR OS SEGUNTES PASSOS:
    - UTILIZAR A PALAVRA-CHAVE "*FUNCTION*";
    - DAR UM NOME À FUNÇÃO;
    - DEFINIR OS PARÂMETROS EM UMA LISTA DELIMITADA POR VÍRGULAS E PARÊNTESES;
    - FAZER O CORPO DA FUNÇÃO, QUE DEVE FICAR ENTRE CHAVES. NESSA PARTE, TEREMOS TODO O CÓDIGO QUE SERÁ EXECUTADO QUANDO UMA FUNÇÃO FOR CHAMADA.

CONCEITOS DE FUNÇÕES BÁSICAS

- INSTRUÇÕES DE FUNÇÃO

EXEMPLO:

```
function traceParameter(aParam:String)
        {
        Trace(aParam);
        }
traceParameter("Hello");
```

CONCEITOS DE FUNÇÕES BÁSICAS

- EXPRESSÕES DE FUNÇÃO
  - A SEGUNDA FORMA DE DECLARAR UMA FUNÇÃO É USAR UMA INSTRUÇÃO DE ATRIBUIÇÃO COM UMA EXPRESSÃO DE FUNÇÃO, QUE ÀS VEZES TAMBÉM É CHAMADA DE LITERAL OU ANÔNIMA. ESTE MÉTODO É MAIS DETALHADO E AMPLAMENTE USADO NAS VERSÕES ANTERIORES DO ACTIONSCRIPT. DEVEM SER FEITOS OS SEGUINTES PASSOS:

    1) UTILIZAR A PALAVRA-CHAVE "*VAR*";

    2) COLOCAR O NOME DA FUNÇÃO;

    3) ADICIONAR O OPERADOR DOIS-PONTOS (:);

CONCEITOS DE FUNÇÕES BÁSICAS

- EXPRESSÕES DE FUNÇÃO

4) CLASSE "*FUNCTION*" PARA INDICAR OS TIPOS DE DADOS;

5) OPERADOR DE ATRIBUIÇÃO *(=)*;

6) A PALAVRA-CHAVE "*FUNCTION*";

7) DEFINIR OS PARÂMETROS EM UMA LISTA DELIMITADA POR VÍRGULAS E PARÊNTESES;

8)FAZER O CORPO DA FUNÇÃO, QUE DEVE FICAR ENTRE CHAVES. NESSA PARTE, TEREMOS TODO O CÓDIGO QUE SERÁ EXECUTADO QUANDO UMA FUNÇÃO FOR CHAMADA.

CONCEITOS DE FUNÇÕES BÁSICAS

- EXPRESSÕES DE FUNÇÃO
  - EXEMPLO:

```
var traceParameter:Function = function(aParam:Sring)
        {
                Trace(aParam);
        }
traceParameter(“hello”);
```

CONCEITOS DE FUNÇÕES BÁSICAS

- ## EXPRESSÕES DE FUNÇÃO

  OBSERVE QUE UM NOME DE FUNÇÃO NÃO É ESPECIFICADO DA MESMA FORMA QUE EM UMA INSTRUÇÃO DE FUNÇÃO. OUTRA DIFERENÇA IMPORTANTE AS DUAS É QUE UMA EXPRESSÃO DE FUNÇÃO NÃO É SUFICIENTE COMO UMA INSTRUÇÃO DE FUNÇÃO. NO EXEMPLO SEGUINTE, TEMOS UMA FUNÇÃO DE EXPRESSÃO ATRIBUÍDA A UMA MATRIZ:

```
var traceArray:Array = new Array();
traceArray[0] = function (aParam: string)
{
        Trace(aParam);
};
traceArray[0](“hello”);
```

CONCEITOS DE FUNÇÕES BÁSICAS

- ESCOLHA ENTRE INSTRUÇÕES E EXPRESSÕES

SEMPRE UTILIZE UMA INSTRUÇÃO DE FUNÇÃO, POIS ELAS SÃO MENOS DETALHADAS E FORNECEM UMA EXPERIÊNCIA MAIS CONSISTENTE, SÃO MAIS FÁCEIS DE LER E TORNAM O CÓDIGO MAIS CONCISO, ALÉM DE SEREM MENOS CONFUSAS DO QUE AS EXPRESSÕES DE FUNÇÃO, QUE REQUEREM O USO DAS PALAVRAS-CHAVE "*VAR*" E "*FUNCTION*". PORÉM, CASO ALGUMA CIRCUNSTÂNCIA ESPECÍFICA EXIJA O USO DE UMA EXPRESSÃO, UTILIZE-A.

CONCEITOS DE FUNÇÕES BÁSICAS

- RETORNANDO VALORES DE FUNÇÕES

PARA RETORNAR UM VALOR, É NECESSÁRIO UTILIZAR "*RETURN*". EM SEGUIDA, COLOQUE A EXPRESSÃO OU VALOR QUE DESEJA RETORNAR.

```
function doubleNum(baseNum:int):int
{
    return (baseNum * 2);
}
```

CONCEITOS DE FUNÇÕES BÁSICAS

- ## FUNÇÕES ANINHADAS

ÚTIL PARA DECLARAR UMA OU MAIS FUNÇÕES DENTRO DE OUTRA. ESTÁ DISPONÍVEL APENAS DENTRO DA FUNÇÃO PAI, A MENOS QUE UMA REFERÊNCIA A ELA SEJA TRANSMITIDA AO CÓDIGO EXTERNO.

```
function getNameAndVersion():String
{
    function getVersion():String
    {
        return "10";
    }
    function getProductName():String
    {
        return "Flash Player";
    }
    return (getProductName() + " " + getVersion());
}
trace(getNameAndVersion());
```

REFERÊNCIAS

- HTTPS://PT.WIKIPEDIA.ORG/WIKI/ACTIONSCRIPT
- HTTP://WWW.ADOBE.COM/DEVNET/ACTIONSCRIPT/LEARNING.HTML
- HTTP://HELP.ADOBE.COM/PT_BR/ACTIONSCRIPT/3.0_PROGRAMMINGAS3/WS5B3CCC516D4FBF351E63E3D118A9B90204-8000.HTML
- HTTP://HELP.ADOBE.COM/PT_BR/ACTIONSCRIPT/3.0_PROGRAMMINGAS3/FLASH_AS3_PROGRAMMING.PDF
- HTTP:// WWW.ADOBE.COM/DEVNET/ACTIONSCRIPT