UNIVERSIDAD POLITECNICA DE VALENCIA

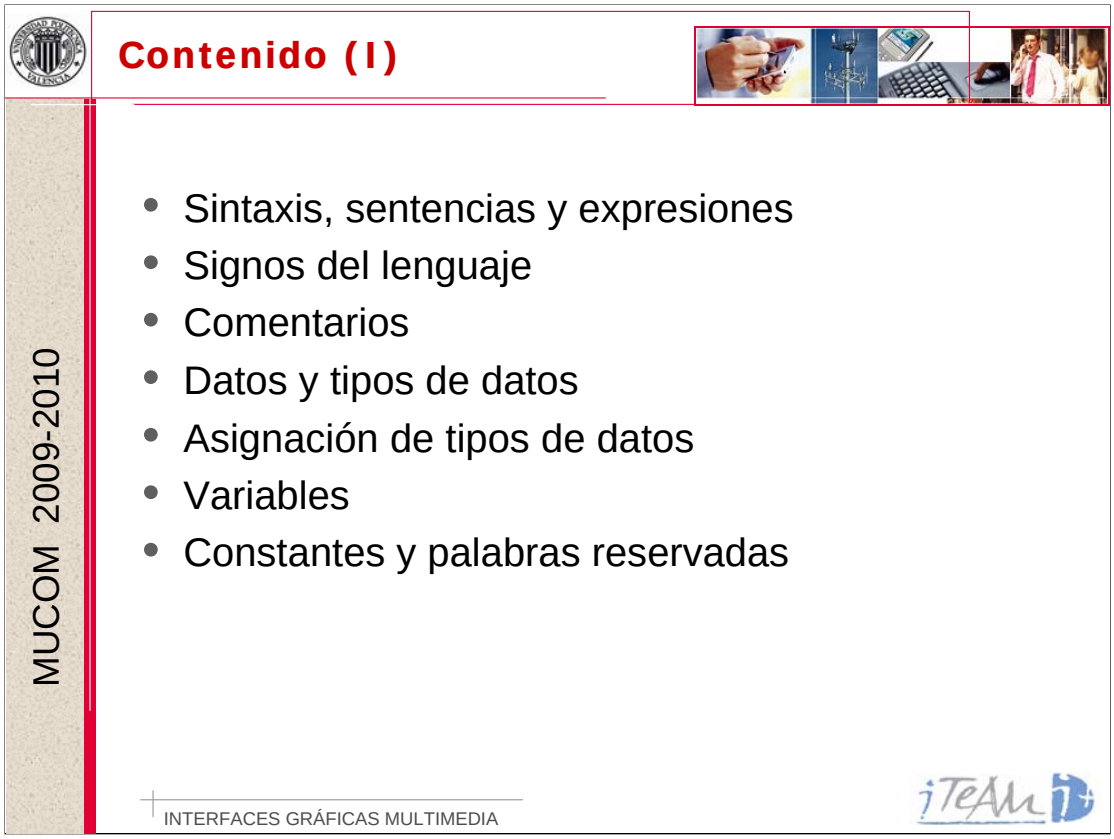
MASTER EN COMUNICACIONES Y DESARROLLO DE SERVICIOS MÓVILES

2009-2010

INTERFACES GRÁFICAS MULTIMEDIA

El lenguaje ActionScript

iTeAM Instituto de Telecomunicaciones y Aplicaciones Multimedia



Este tema presenta la esencia del lenguaje de programación ActionScript, en concreto la versión 2 – es la que se suele incluir de fábrica actualmente en los dispositivos móviles. Una vez que se conozcan estos fundamentos, estaremos preparados para desarrollar aplicaciones con Flash Lite 2.x y 3.x. Si queremos desarrollar aplicaciones para Flash Lite 1.x hay que tener en cuenta unas cuantas diferencias en cuanto a la sintaxis del código, las cuales no se detallarán en este curso por ser muy obsoletas.

Contenido (II)

- Condiciones
- Bucles
- Matrices
- Operadores
- Funciones
- Gestión de eventos

MUCOM 2009-2010

Contenido (I)

MUCOM 2009-2010

- Sintaxis, sentencias y expresiones ►
- Signos del lenguaje
- Comentarios
- Datos y tipos de datos
- Asignación de tipos de datos
- Variables
- Constantes y palabras reservadas

Sintaxis, sentencias y expresiones

MUCOM 2009-2010

- Sintaxis: gramática y ortografía de un lenguaje
  El conjunto de reglas y directrices para programar.
- Expresión: combinación (válida) de símbolos
  Puede estar formada por operadores y operandos, valores, funciones y procedimientos. Normalmente es una sola línea de código.
- Sentencia: código de instrucción
  Puede tener una o varias líneas de código.

El lenguaje ActionScript está formado por clases incorporadas. La sintaxis se refiere a la gramática y la ortografía de un lenguaje: es un conjunto de reglas y directrices para programar correctamente.

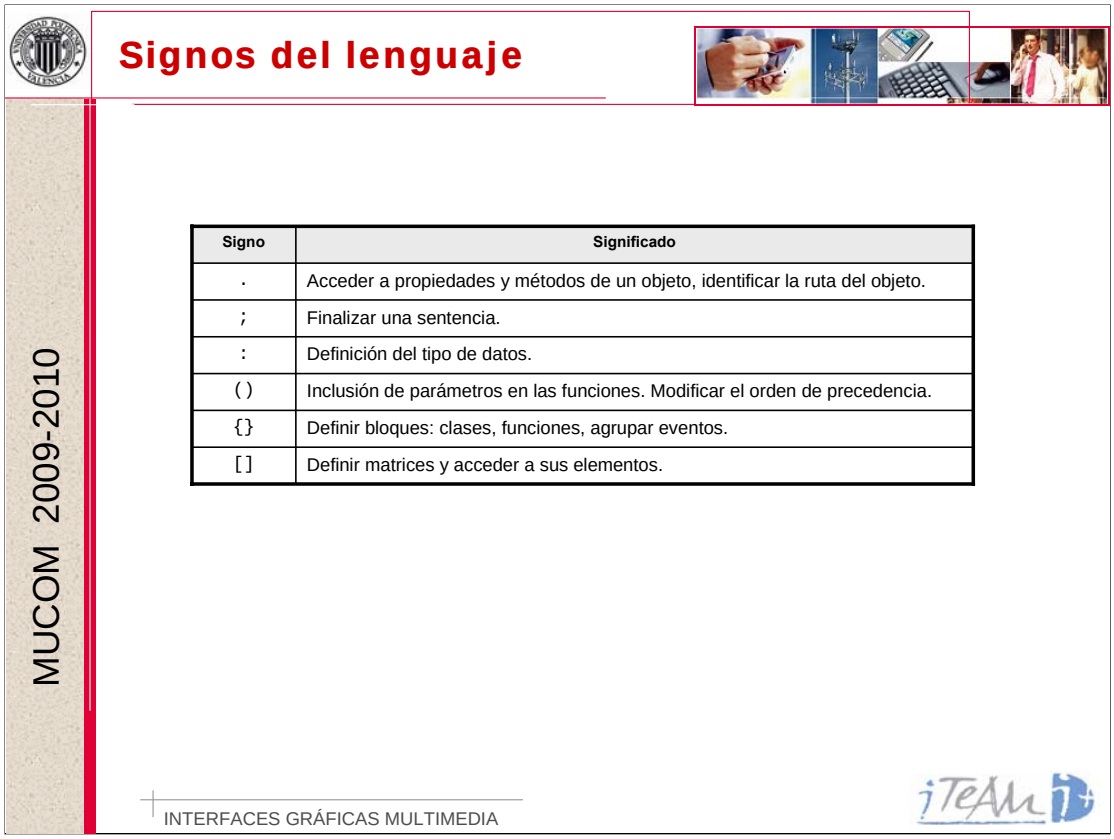
Una sentencia es una instrucción para algo concreto. Por ejemplo, podemos utilizar una sentencia condicional para determinar si algo es verdadero o si existe para, posteriormente, ejecutar alguna acción.

Las expresiones, a diferencia de las sentencias, son cualquier combinación válida de símbolos de ActionScript que representan un valor. Las expresiones tienen valores, mientras que los valores y las propiedades tienen tipos. Una expresión puede estar formada por operadores y operandos, valores, funciones y procedimientos. La expresión sigue las reglas de prioridad y asociatividad de ActionScript. Normalmente, Flash Player interpreta la expresión y luego devuelve un valor que podemos utilizar en la aplicación.

Contenido (I)

MUCOM 2009-2010

- Sintaxis, sentencias y expresiones
- Signos del lenguaje ►
- Comentarios
- Datos y tipos de datos
- Asignación de tipos de datos
- Variables
- Constantes y palabras reservadas



El tipo más habitual de signos son el punto, el punto y coma, los dos puntos, los paréntesis, las llaves y los corchetes, cada uno con un significado específico.

En ActionScript, el operador de punto (`.`) permite acceder a propiedades y métodos que pertenecen a un objeto o instancia. También se utiliza para identificar la ruta de destino de una instancia, una variable, una función o un objeto. Para controlar un MovieClip, por ejemplo, es preciso especificar una ruta de destino. Las rutas de destino son direcciones jerárquicas de nombres de instancias. Para especificar una ruta de destino hay que asignar un nombre de instancia al MovieClip.

El uso del punto y coma (`;`) para terminar una sentencia permite colocar más de una sentencia en una misma línea, pero, al hacerlo, normalmente el código resulta más difícil de leer.

Los dos puntos (`:`) se utilizan en el código para asignar tipos de datos a las variables. Para asignar un tipo de datos específico a un elemento, hay que especificar la palabra clave `var` antes de dicha asignación (ej: `var my_mc:MovieClip`).

Los paréntesis (`( )`) se usan para modificar el orden de precedencia de ActionScript o para hacer más legibles las sentencias. Asimismo, al definir una función, los parámetros (si tiene) se colocan dentro de los paréntesis.

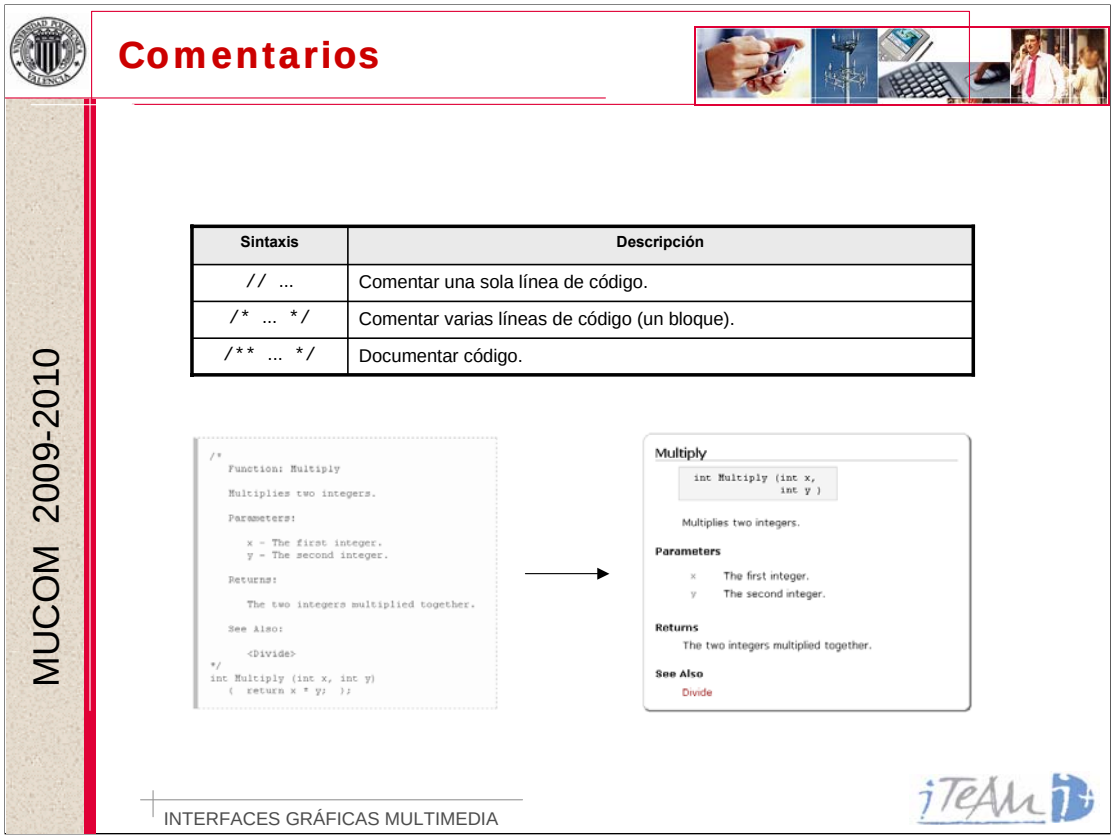
Las llaves (`{}`) permiten agrupar eventos, definiciones de clases y funciones de ActionScript en bloques. La llave inicial se suele colocar en la misma línea que la declaración, aunque también se puede poner en la línea que sigue a la declaración, dependiendo de la convención que use el equipo de programación.

Los corchetes (`[]`) se emplean normalmente para inicializar vectores y matrices con determinados valores y acceder a ellos mediante su número de índice. Más adelante se detallan las matrices en ActionScript.

Contenido (I)

MUCOM 2009-2010

- Sintaxis, sentencias y expresiones
- Signos del lenguaje
- Comentarios ►
- Datos y tipos de datos
- Asignación de tipos de datos
- Variables
- Constantes y palabras reservadas



Los comentarios son una forma de realizar anotaciones en el código que el compilador no evalúa. Los comentarios documentan las decisiones que se toman con respecto al código y hacen que resulte más comprensible. El uso de comentarios ayuda a recordar decisiones importantes de codificación y es de gran ayuda para otras personas que lean el código. Los comentarios deben explicar claramente el propósito del código y no simplemente traducirlo.

Los comentarios pueden tener cualquier longitud sin que ello afecte al tamaño del archivo exportado y no es necesario que sigan las reglas de las palabras clave y de la sintaxis de ActionScript.

Los comentarios de una sola línea suelen utilizarse para explicar un pequeño fragmento de código. Para indicar que una línea es un comentario, se empieza a escribir con dos barras inclinadas (//).

Los comentarios de varias líneas, también llamados comentarios en bloque, se usan normalmente para describir archivos, estructuras de datos y métodos. Normalmente se colocan al principio de un archivo y en aquellas secciones de código que necesiten más explicación.

Los comentarios dentro de las clases permiten documentarlas con el fin de que los desarrolladores comprendan mejor el contenido. Normalmente se empiezan todos los archivos de clase con un comentario que indica el nombre de clase, el número de versión, la fecha y el copyright. La sintaxis para documentar código AS más extendida es JavaDoc (http://es.wikipedia.org/wiki/Javadoc), si bien para AS2 también goza de popularidad la sintaxis NaturalDocs (http://www.naturaldocs.org).

Contenido (I)

MUCOM 2009-2010

- Sintaxis, sentencias y expresiones
- Signos del lenguaje
- Comentarios
- Datos y tipos de datos ►
- Asignación de tipos de datos
- Variables
- Constantes y palabras reservadas

Datos y tipos de datos

MUCOM 2009-2010

Array	Boolean	Button	Color	Date
Error	Function	Loadvars	MovieClip	MovieClipLoader
Number	Object	SharedObject	Sound	String
TextField	TextFormat	XML	XMLNode	XMLSocket
XMLUI	*undefined*	*null*	*Void*	

Los datos son números, cadenas u otra “información” que se puede manipular. Ponemos “información” entre comillas porque, como dijo alguien, técnicamente los datos no son información hasta que se procesan. Un tipo de datos describe el dato en sí mismo y los tipos de operaciones que pueden realizarse con él. Los datos se almacenan en variables.

Un valor simple (o tipo de dato simple) es un valor que ActionScript almacena en el nivel más bajo de abstracción, lo que significa que las operaciones con tipos de datos simples son generalmente más rápidas y eficientes que las operaciones realizadas con tipos de datos complejos. Los siguientes tipos de datos definen un conjunto de uno o varios valores simples: `Boolean` (booleano), `null` (nulo), `Number` (número), `String` (cadena) y `undefined` (no definido).

Un valor complejo (o tipo de datos complejo) es un valor menos abstracto y que hace referencia a los valores simples. Estos se denominan con frecuencia tipos de datos de referencia. Los valores complejos pertenecen al tipo de datos `Object` (objeto) o a un tipo de datos basado en el tipo de datos `Object`. Entre los tipos de datos que definen conjuntos de valores complejos se encuentran `Array` (matriz), `Date` (fecha), `Error` (error), `Function` (función) y `XML`.

Contenido (I)

MUCOM 2009-2010

- Sintaxis, sentencias y expresiones
- Signos del lenguaje
- Comentarios
- Datos y tipos de datos
- Asignación de tipos de datos ►
- Variables
- Constantes y palabras reservadas

Asignación de tipos de datos

MUCOM 2009-2010

- Un dato...
  - En AS1 puede ser de cualquier tipo
    Versatilidad contra eficiencia.
  - En AS2 puede (y debe) ser de un solo tipo
    Declaración mediante Strict Data Typing: `var nombre:tipo = valor;`
  - En AS3 sólo puede ser de un tipo
    Eficiencia contra versatilidad (aunque existe el tipo *).

```
var lastName:String = "Pérez Sánchez";
var score:Number = 153.48;
var isNew:Boolean = false;
var as3Only:*;
```

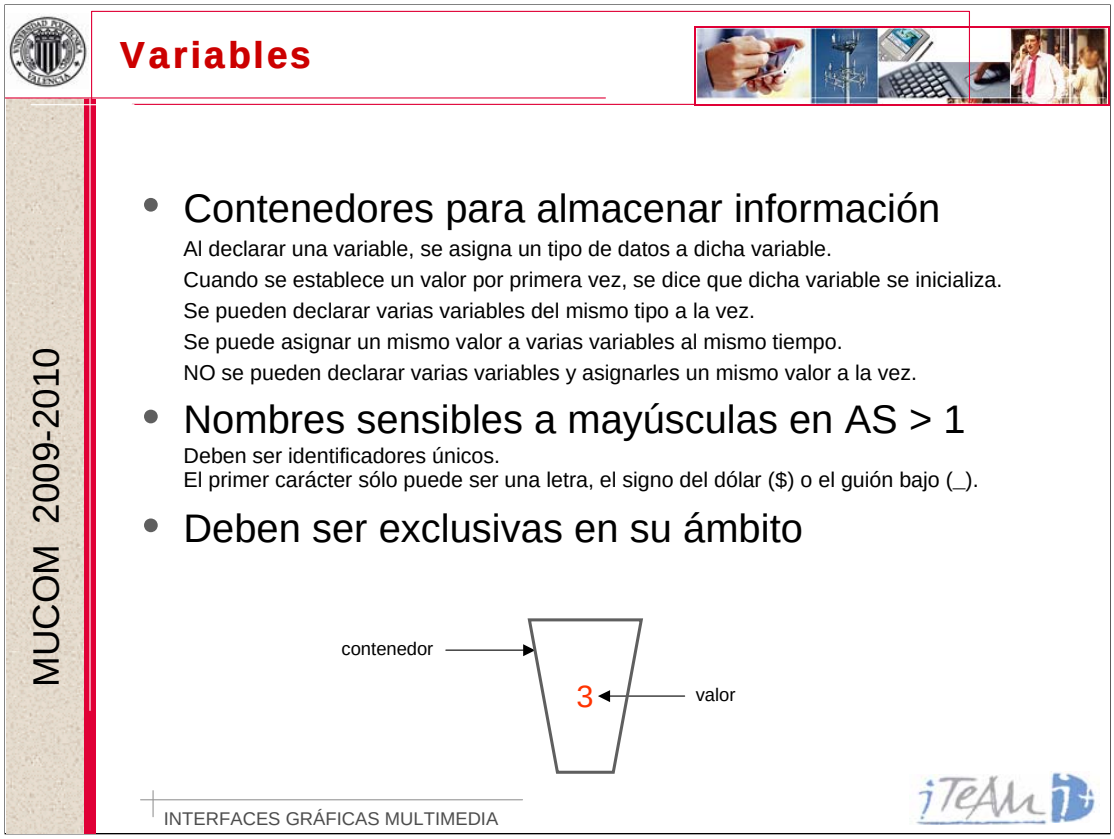

Las variables se utilizan para incluir valores en el código. Una variable asignada a un tipo de datos sólo puede contener un valor entre el conjunto de valores disponibles para dicho tipo de datos. Se puede (y se debe) declarar de forma explícita el tipo de objeto de una variable al crearla, lo que recibe el nombre de *Strict Data Typing*. Se recomienda usar Strict Data Typing siempre que sea posible ya que agrega funcionalidad complementaria al código, lo cual acelera el proceso de ejecución, y genera errores en la compilación, si los hubiera, mostrándolos en el panel de Salida.

Para una definición estricta del tipo de variables, hay que declararlas con la palabra clave `var`. En ocasiones, Strict Data Typing se conoce como la *comprobación de tipos* al compilar una variable.

Contenido (I)

MUCOM 2009-2010

- Sintaxis, sentencias y expresiones
- Signos del lenguaje
- Comentarios
- Datos y tipos de datos
- Asignación de tipos de datos
- Variables ►
- Constantes y palabras reservadas

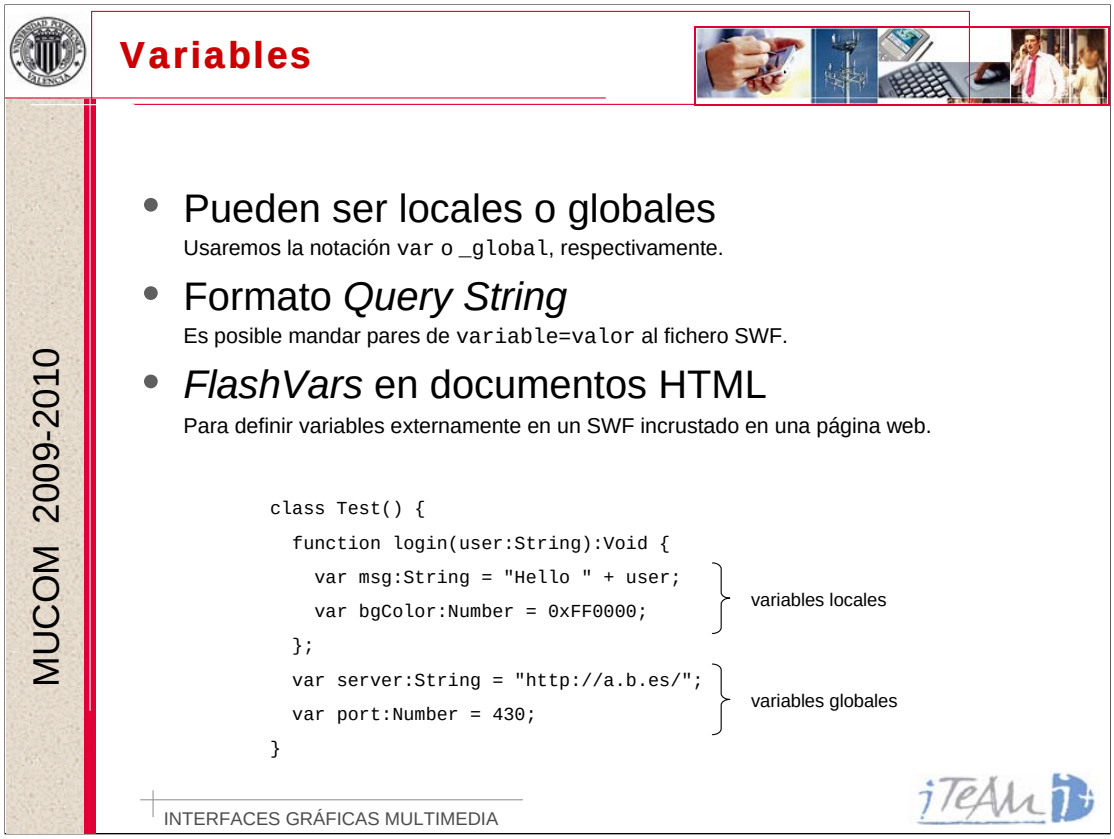


Una variable es un contenedor que almacena información. Se pueden declarar variables en un fotograma de la línea de tiempo, directamente en un objeto o en un archivo de clase externo. Al declarar una variable, se asigna un tipo de datos a la variable.

El valor que contiene una variable antes de establecer su valor se denomina valor predeterminado. Cuando se establece su valor por primera vez, se dice que la variable se inicializa. Si se declara una variable pero no se establece su valor, dicha variable adopta de manera predeterminada el valor `undefined`. Se pueden declarar varias variables del mismo tipo a la vez, incluso se puede asignar un mismo valor a varias variables al mismo tiempo, pero no se pueden declarar varias variables y asignarles un mismo valor a todas a la vez.

Una variable debe ser un identificador. El primer carácter de un identificador sólo puede ser una letra, el signo del dólar ($) y el guión bajo (_). No se puede comenzar el nombre con un número, pero sí puede usarse en caracteres posteriores. Los identificadores deben ser únicos y son sensibles a mayúsculas. Obviamente, tampoco se pueden emplear nombres reservados, como por ejemplo `break`, `switch` o `if`. Una variable, además, debe ser exclusiva en su ámbito. El ámbito de una variable se refiere al área en la que se conoce la variable (la zona en la que está definida) y desde la cual se puede hacer referencia a dicha variable. Esta zona puede estar dentro de una determinada línea de tiempo o dentro de una función, o podría conocerse globalmente en toda la aplicación.

Las variables de línea de tiempo están disponibles para cualquier script de dicha línea de tiempo. Para declarar variables de línea de tiempo, se usa la sentencia `var` en cualquier fotograma de la línea de tiempo. La variable está disponible para dicho fotograma y todos los fotogramas posteriores.



Al utilizar la sentencia `var` dentro de un bloque de función, se declaran las variables locales. Al declarar una variable local dentro de un bloque de función (también conocido como definición de función), se define dentro del ámbito de la función y *caduca* al final de dicha función. Por consiguiente, la variable local sólo existe dentro de esa función. Los parámetros pasados a una función son considerados variables locales.

Las definiciones globales de funciones y variables son visibles para todas las líneas de tiempo y todos los ámbitos del documento. Para declarar una variable o función de ámbito global, se escribe el identificador `_global` delante del nombre de la variable, en este caso nunca se emplea la sintaxis `var`.

Si queremos pasar valores de una página HTML a un fichero SWF, los valores transferidos llevan el formato *Query String* (cadena de consulta), también llamados "variables con codificación URL". Estas variables se envían separando pares `nombre=valor` mediante un carácter ampersand (&). Si hacemos la carga de variables desde una dirección de Internet, la primera variable debe empezar con el signo de interrogación (?), y el resto con el ampersand. Por ejemplo:

```
sample.swf?userid=3447575&verified=1
```

Cuando se incrusta contenido Flash en un documento HTML, la notación a emplear es esta:

```
<object type="application/x-shockwave-flash" data=videoplayer.swf" width="400"
height="225">
  <param name="movie" value="videoplayer.swf">
  <param name="flashVars"
   value="clip_id=7231932&amp;server=vimeo.com&amp;show_title=1&amp;fullscreen=1">
  <param name="allowFullScreen" value="true">
  <p>Sample video.</p>
</object>
```

Contenido (I)

MUCOM 2009-2010

- Sintaxis, sentencias y expresiones
- Signos del lenguaje
- Comentarios
- Datos y tipos de datos
- Asignación de tipos de datos
- Variables
- Constantes y palabras reservadas ►

Constantes y palabras reservadas

MUCOM 2009-2010

- Constantes: propiedades con un valor fijo

  Su valor no cambia en todo el tiempo de ejecución.

  Permiten leer el código con mayor facilidad y ayudan a evitar errores de programación.

  Cada clase suele tener definidas unas constantes (`$version`, `Key.ENTER`, `Math.PI` ...).

- No se pueden usar palabras reservadas como identificadores

  Algunos ejemplos:

`break`	`case`	`catch`	`class`	`continue`
`default`	`else`	`extends`	`finally`	`for`
`function`	`import`	`instanceof`	`new`	`private`
`return`	`switch`	`static`	`this`	`throw`
`try`	`typeof`	`var`	`while`	`with`

Las constantes y las palabras clave constituyen la base de la sintaxis de ActionScript. Las constantes son propiedades con un valor fijo que no puede alterarse, son valores que no cambian en ninguna parte de una aplicación.

Flash Lite incluye diversas constantes predefinidas que contribuyen a simplificar el desarrollo de aplicaciones. Un ejemplo de constantes lo podemos encontrar en la clase `Key`, que incluye numerosas propiedades, como `Key.ENTER` o `Key.PGDN`. Si basamos nuestro código en constantes, no hará falta recordar que los valores de código de las teclas Intro y AvPág son 13 y 34. La utilización de valores constantes no hace que el desarrollo o la depuración resulten más sencillos, pero sí permite a otros desarrolladores leer el código con mayor facilidad.

Las palabras reservadas de ActionScript se utilizan para escribir el código, de modo que no se pueden utilizar como identificadores; por ejemplo, como nombres de variables, de funciones o de etiquetas. Asimismo, todos los nombres de clases, componentes o interfaces del *core* de ActionScript son reservados y no pueden utilizarse como identificadores en el código.

Contenido (II)

MUCOM 2009-2010

- Condiciones ►
- Bucles
- Matrices
- Operadores
- Funciones
- Gestión de eventos

Condiciones

MUCOM 2009-2010

- Determinan si algo es verdadero o se cumple

1.
```
if (condición) { //sentencias; }
```

2a.
```
if (condición) { //sentencias si true; } else { //sentencias si false; }
```
2b.
```
(condición) ? //sentencia si true : //sentencia si false;
```

3.
```
if (condition1) { //sentencias si condition1 es true;
} else if (condition2) { //sentencias si condition2 es true;
} else { //sentencias en otro caso;
}
```

4.
```
switch (condition) {
 case A :
   // sentencias;
   // si no hay break se pasa al siguiente caso
 case B :
   // sentencias;
   break;
 default :
   // sentencias en otro caso;
   break;
}
```


Las condiciones permiten determinar si algo es verdadero o existe y, seguidamente, ejecutar las acciones que se especifiquen.

Se emplea la sentencia `if` cuando se desea ejecutar una serie de sentencias en función de si una determinada condición es verdadera (`true`) o no (`false`).

Se usa la sentencia `if...else` para comprobar una condición y, seguidamente, ejecutar un bloque de código si dicha condición existe, o ejecutar un bloque de código alternativo si dicha condición no existe. En el caso de condiciones `if...else` que llevan sólo una sentencia para `true` y otra para `false`, se puede emplear esta notación alternativa:

`(`*`condición`*`) ? `*`sentencia si true`*` : `*`sentencia si false`*`;`

Ojo a la terminación de esta línea en punto y coma, cosa que no se ha hecho en la *sentencia si true* porque la línea de código debe continuar con los dos puntos hasta la *sentencia si false*.

Se utiliza la sentencia condicional `if...else if...` para comprobar distintas condiciones.

Cuando se desean evaluar muchos valores de una condición, lo habitual es emplear la sentencia `switch`. Al igual que la sentencia `if`, la sentencia `switch` prueba una condición y ejecuta sentencias si la condición devuelve un valor `true`.

La sentencia `break` ordena que se omita el resto de sentencias existentes en ese bloque `case` y que salte a la primera sentencia que vaya a continuación de la sentencia `switch`. Si un bloque `case` no contiene una sentencia `break`, se producirá una condición conocida como de "paso al siguiente caso". En esta situación, la siguiente sentencia `case` también se ejecuta hasta que se encuentra una sentencia `break` o termina la sentencia `switch`. Todas las sentencias `switch` deben incluir un caso `default` (predeterminado). El caso `default` siempre suele ser el último caso de una sentencia `switch` y también debe incluir una sentencia `break` para evitar un error de paso al siguiente `case` (si se añadiera otro caso en un futuro).

Contenido (II)

MUCOM 2009-2010

- Condiciones
- Bucles ►
- Matrices
- Operadores
- Funciones
- Gestión de eventos

Bucles

- Permiten repetir una serie de sentencias cuando se cumple una condición

1.
```
for (inicio; condición; paso) { //sentencias; }
```

2.
```
var myObj:Object = {x:20, y:30};
for (var i:String in myObj) { trace(i + " = " + myObj[i]); }
```

3.
```
var i:Number = 0;
while (i < 5) {
  trace(i);
  i++; //si no se actualiza el valor de i, el bucle es infinito;
}
```

4.
```
var i:Number = 2;
do {
  trace(i);
  i++;
} while (i < 5);
```

MUCOM 2009-2010

Los bucles permiten repetir una serie de sentencias cuando una condición se evalúa a `true`. Existen cuatro tipos de bucles en ActionScript: los bucles `for`, los bucles `for...in`, los bucles `while` y los bucles `do...while`. Cada tipo de bucle se comporta de forma ligeramente distinta, por lo que cada uno de ellos resulta útil en una determinada situación.

La mayoría de las reproducciones indefinidas utilizan algún tipo de contador para controlar cuántas veces se ejecutan. Cada ejecución se denomina repetición o iteración. Lo normal es declarar una variable y escribir una sentencia que incremente o disminuya el valor de la variable cada vez que se ejecute la reproducción.

Los bucles `for` permiten repetir una variable para un intervalo de valores específico. Resultan útiles cuando se conoce exactamente el número de veces que es necesario repetir una serie de sentencias. Un bucle `for` repite una acción mediante un contador incorporado. El contador y la sentencia que incrementa el contador son parte de la sentencia `for`.

La sentencia `for...in` se usa para ejecutar en bucle los subniveles de un MovieClip, las propiedades de un objeto o los elementos de una matriz. Los subniveles pueden incluir otros MovieClips, funciones, objetos y variables. Las propiedades de un objeto no se guardan en ningún orden concreto, por lo que aparecen de forma impredecible al usar este tipo de bucle.

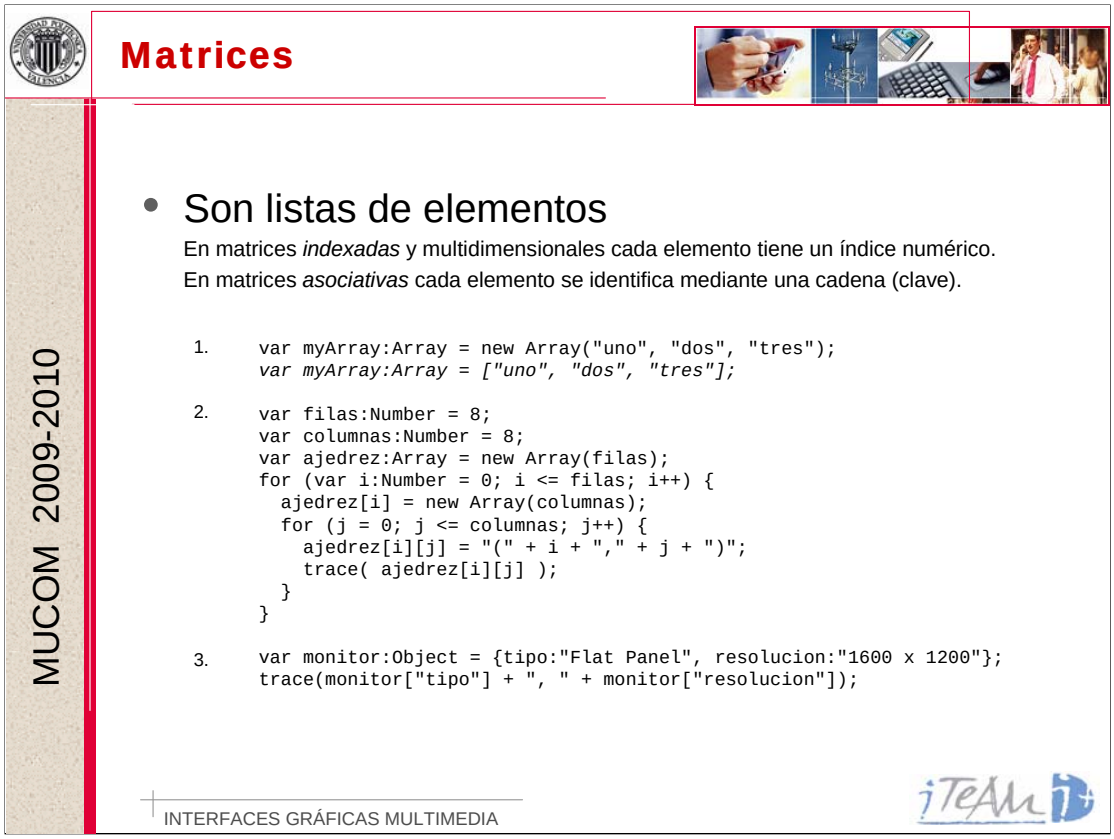
La sentencia `while` se emplea para repetir una acción mientras sea verdadera una condición y, a diferencia del bucle `for`, sin conocer explícitamente el intervalo de valores. Es similar a una sentencia `if` que se repite indefinidamente mientras la condición sea `true`. Un bucle `while` calcula el resultado de una expresión y ejecuta el código en el cuerpo del bucle si la expresión es `true`, antes de regresar para comprobar la condición de nuevo. Cuando la condición dé como resultado `false`, finaliza el bucle.

La sentencia `do...while` se utiliza para crear el mismo tipo de repetición que `while`, pero la expresión se comprueba después de que se ejecute el bloque de código, por lo que siempre se ejecuta al menos una iteración. Las sentencias sólo se ejecutan si la condición da como resultado `true`.

Contenido (II)

MUCOM 2009-2010

- Condiciones
- Bucles
- Matrices ►
- Operadores
- Funciones
- Gestión de eventos



Una Matriz, o `Array`, es un objeto cuyas propiedades se identifican mediante números que representan sus posiciones en la estructura. Básicamente se trata de una lista de elementos. Es importante recordar que no todos los elementos de una matriz tienen que pertenecer al mismo tipo de datos. Podemos combinar números, fechas, cadenas, objetos e incluso añadir una matriz anidada en cada índice de la matriz. Los tipos de matrices que acepta ActionScript son las matrices indexadas, las matrices multidimensionales y las matrices asociativas.

Las matrices indexadas almacenan uno o varios valores, a los cuales se accede mediante el índice, esto es, el número que representa su posición en la matriz. El primer número de índice siempre es `0` y aumenta una unidad por cada elemento posterior que añada a la matriz. Una matriz indexada se crea llamando al constructor de la clase `Array` o inicializando la matriz con el literal de matriz `[]`.

Las matrices multidimensionales son matrices anidadas, esto es, matrices de matrices, las cuales pueden concebirse visualmente como cuadrículas. Por ejemplo, un tablero de ajedrez es una cuadrícula de ocho columnas y ocho filas; puede modelarse como una matriz fila con ocho elementos, cada uno de los cuales es, a su vez, una matriz columna que contiene ocho elementos.

Las matrices asociativas utilizan claves en lugar de un índice numérico para organizar los valores almacenados. Cada clave es una cadena exclusiva y está asociada y se utiliza para acceder a un valor. El tipo de datos de dicho valor puede ser `Number`, `Array`, `Object`, etc. Al crear código para localizar un valor asociado a una clave, se estará indexando o realizando una consulta. Las matrices asociativas están formada por claves y valores sin ordenar.

La asociación entre una clave y un valor se conoce normalmente como su vinculación. Por ejemplo, una agenda de contactos podría considerarse una matriz asociativa en la que los nombres son las claves y los teléfonos y/o direcciones de correo electrónico son los valores.

Contenido (II)

- Condiciones
- Bucles
- Matrices
- Operadores ►
- Funciones
- Gestión de eventos

Operadores

MUCOM 2009-2010

Unarios	`++`	Incremento	**Relacionales**	`>`	Mayor que
	`--`	Decremento		`<`	Menor que
	`!`	Negación		`<=`	Mayor o igual que
	`typeof`	Informa del tipo		`>=`	Menor o igual que
	`void`	Valor no definido		`instaceof`	Comprueba el tipo de dato
Multiplicativos	`*`	Multiplicación		`in`	Comprueba propiedades
	`/`	División	**Asignación**	`=`	Asignación
	`%`	Módulo		`*=`	Asignación de multiplicación
Aditivos	`+`	Suma		`/=`	Asignación de división
	`-`	Resta		`%=`	Asignación de módulo
Igualdad	`==`	Igualdad		`+=`	Asignación de suma
	`!=`	Desigualdad		`-=`	Asignación de resta
	`===`	Igualdad estricta	**Lógicos**	`&&`	AND lógico
	`!==`	Desigualdad estricta		`||`	OR lógico

Los operadores son caracteres que especifican cómo combinar, comparar o cambiar los valores de una expresión. Una expresión ya vimos que era cualquier sentencia para la que se puede calcular el resultado y que devuelve un valor.

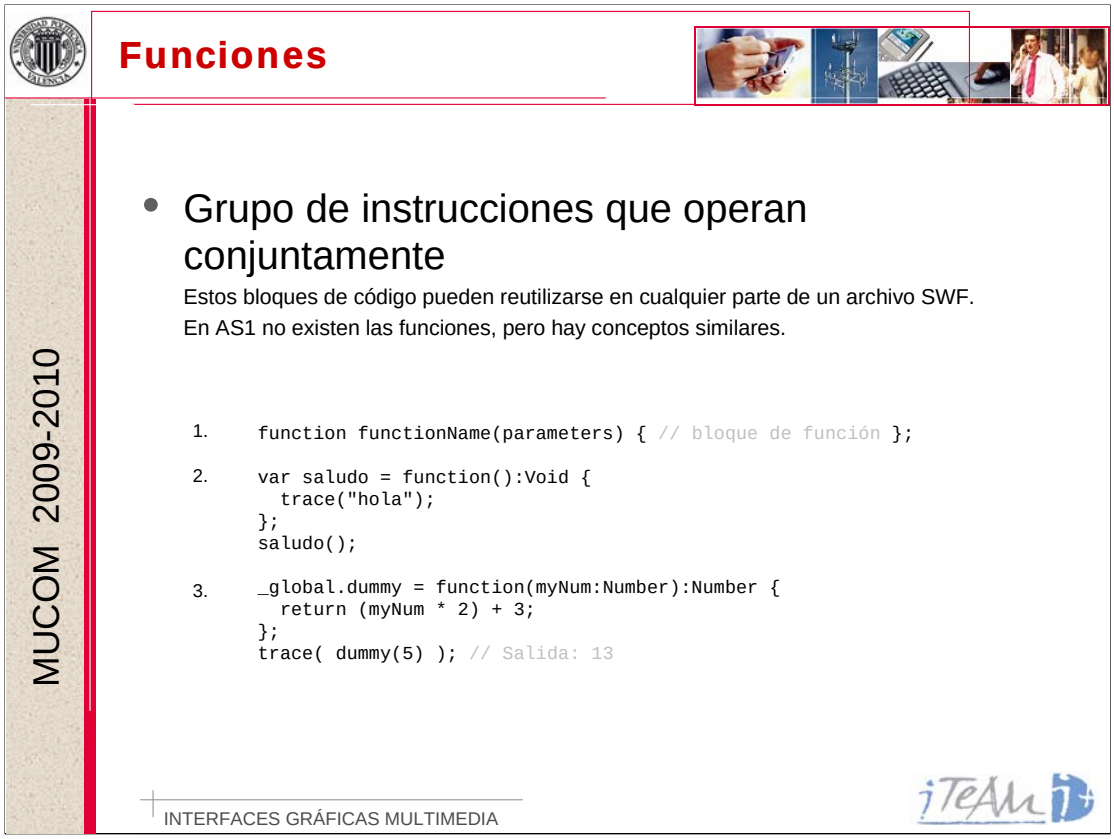
Al utilizar dos o más operadores en una sentencia, algunos operadores tienen precedencia sobre otros. La precedencia y asociatividad de los operadores determina el orden en que se procesan los mismos. ActionScript tiene una jerarquía que determina qué operadores se ejecutan antes que otros, la cual puede consultarse en la documentación oficial y no es el objeto de este curso. Si queremos estar seguros de que cierta operación se procese antes que otra(s), la encerraremos entre paréntesis.

Los operadores se clasifican en unarios (los que utilizan un solo operando), multiplicativos, aditivos, relacionales, de igualdad, de asignación, lógicos y en modo bit. Estos últimos manipulan internamente los números de coma flotante para tratarlos como enteros de 32 bits.

Contenido (II)

MUCOM 2009-2010

- Condiciones
- Bucles
- Matrices
- Operadores
- Funciones ►
- Gestión de eventos



Los métodos y funciones son bloques de código que pueden volver a utilizarse en cualquier parte de un archivo SWF. Podemos escribir funciones en el archivo FLA o en un archivo AS (código ActionScript externo) y, seguidamente, llamar a la función desde cualquier lugar de los documentos. Sólo podemos hacer referencia a una función siempre que la función se haya definido en el fotograma actual o anterior.

El formato estándar para declarar funciones con nombre es el siguiente:

`function nombre(argumentos):tipo_valor_salida { bloque de función };`

También podemos crear una función sin nombre que hace referencia a ella misma; lo que se conoce como función anónima. Las funciones anónimas se utilizan habitualmente al manipular controladores de eventos (en los denominados *callbacks*).

Al igual que las variables, las funciones están asociadas a la línea de tiempo del clip de película que las define y, para llamarlas, se debe utilizar una ruta de destino. Para declarar una función global que esté disponible para todas las líneas de tiempo y ámbitos sin utilizar una ruta de destino se emplea el identificador `_global`.

Contenido (II)

MUCOM 2009-2010

- Condiciones
- Bucles
- Matrices
- Operadores
- Funciones
- Gestión de eventos ►

Gestión de eventos

MUCOM 2009-2010

- La aplicación reacciona ante los procesos de interacción de la información

Cada objeto tiene asociado un determinado tipo de eventos.

Están controlados mediante *callbacks* y *listeners*.

1.
```
object.eventMethod = function() { // código de respuesta al evento };

logo_mc.onPress = function():Void {
  trace(this._name + " ha sido pulsado");
};
```

2.
```
var listenerObject:Object = new Object();
listenerObject.eventName = function(eventObj:Object) {
  // código de la función
};
broadcasterObject.addListener(listenerObject);

var tecla:Object = new Object();
tecla.onKeyDown = function(eventObj:Object):Void {
  trace("has pulsado una tecla")
};
Key.addListener(tecla);
```


Cada clase tiene asociado un determinado tipo de eventos. Con el fin de que la aplicación reaccione ante los distintos eventos, podemos utilizar controladores de eventos, es decir, código ActionScript asociado con un objeto y un evento determinados.

Un controlador de eventos se compone de tres partes: el objeto al que se aplica el evento, el nombre del método del controlador de eventos del objeto y la función que se asigna al controlador.

En el primer ejemplo se aplica una función anónima, por lo que sobre el evento actúa única y exclusivamente dicha función anónima. Los listeners son detectores de eventos más genéricos. Los listeners permiten que un objeto, denominado detector, reciba eventos difundidos por otro objeto, denominado difusor. El objeto difusor (teclado, escenario, ratón...) registra el objeto detector que va a recibir los eventos generados por el difusor. Por ejemplo, se puede registrar un objeto para que reciba notificaciones `onResize` del objeto difusor `Stage`. Se pueden registrar varios objetos detectores para que reciban eventos de un único difusor y se puede registrar un único objeto detector para que reciba eventos de varios difusores.