ACTIONSCRIPT (AS)

Dpto. Escultura. UPV.

[usos del lenguaje]

AS

JERARQUIAS DE CLIP

Nos sirven para recorrer las jerarquía del clip que se ha creado.

2 tipos: -**Absoluta**

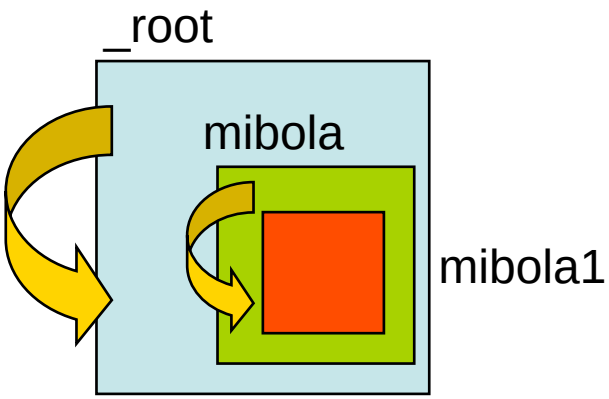
- (Desde el película principal hasta el clip que necesitamos utilizar)

_root.mibola.mibola1._width = 50;

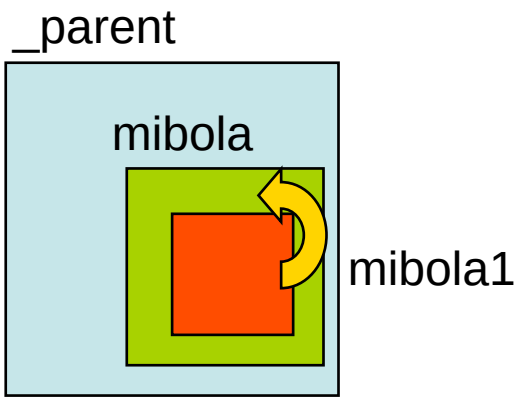
-**Relativa**

- (Desde el clip que necesitamos utilizar hacia el/los superior/es que lo contienen)

_parent._width = 50;



Absoluta



relativa

AS

JERARQUIAS DE CLIP

Nos sirven para recorrer las jerarquía del clip que se ha creado.

2 tipos: -**Absoluta**

- (Desde el película principal hasta el clip que necesitamos utilizar)

```
_root.mibola.mibola1.mibola2._width = 50;
```

-**Relativa**

- (Desde el clip que necesitamos utilizar hacia el/los superior/es que lo contienen)

```
_parent._width = 50;
```

_root Relación jerarquiza del clip desde la película principal

```
_root.mibola.mibola1.mibola2._width = 50;
```

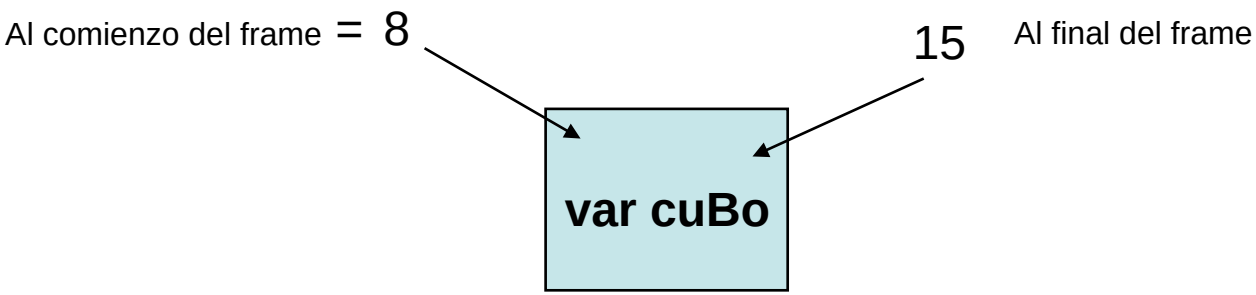
_parent Relación del clip con el anterior

```
this._parent._width = 50;
```

this Lo utilizamos para referirnos a la línea de tiempo del clip con el que estamos trabajando en ese momento

```
this._width = 50;
```

Podemos entenderlos como contenedores de datos fijos o variables (ver tipos de datos en el PP anterior)



- Podemos asignarles un valor ó operar con ellos.

```
cuBo = 10;
cuBo = 10 + 5;
cuBo = cuBito1 + cuBito2
```

Numérico (int)

```
cuBito1 = “lle”;
cuBito2 = “no”;
cuBo = cuBito1 + cuBito2
```

Cadena (String)

AS

VARIABLES

Podemos entenderlos como contenedores de datos fijos o variables (ver tipos de datos en el PP anterior)

Podemos también utilizar variables **GLOBALES**

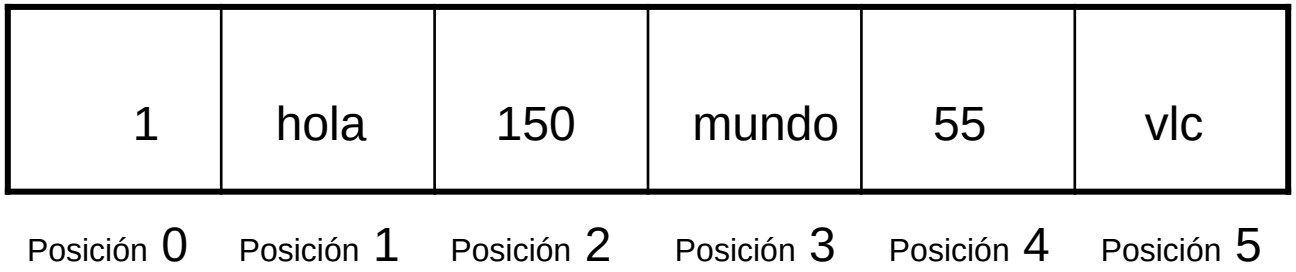
_global.cuBo =5;

identificador

- cuBo será una variable que podremos llamar durante toda la película, ya que al ser global su radio de acción es todo el proyecto.

Una variable sólo puede almacenar un valor en un instante determinado.
Los arrays nos permiten contener varios valores en el mismo instante.

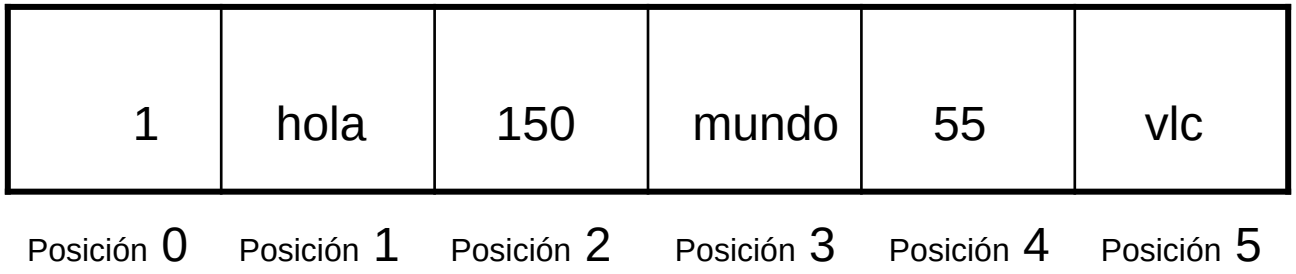
ARRAY - Almacena datos



Este Array contienen 6 contenedores de datos
siempre comienza por 0

Una variable sólo puede almacenar un valor en un instante determinado.
Los arrays nos permiten contener varios valores en el mismo instante.

ARRAY - Almacena datos



Declaración de las varias formas:

Método constructor

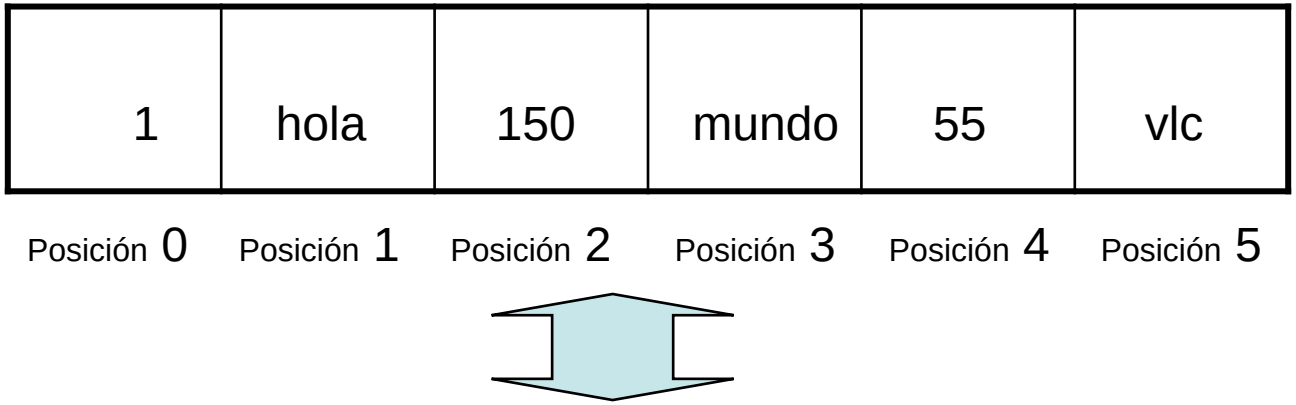
Cubo_almacen = **new Array();**

Cubo_almacen = **new Array(1,”hola”,150,”mundo”,55,”vlc”);**

Cubo_almacen = **[1,”hola”,150,”mundo”,55,”vlc”];**

Una variable sólo puede almacenar un valor en un instante determinado.
Los arrays nos permiten contener varios valores en el mismo instante.

ARRAY - Almacena datos



Declaración de cada contenedor

```
Cubo_almacen = new Array();
Cubo_almacen[0] = 1;
Cubo_almacen[1] = “hola”;
Cubo_almacen[2] = 150;
Cubo_almacen[3] = “mundo”;
Cubo_almacen[4] = 55;
Cubo_almacen[5] = “vlc”;
```

AS

EVENTOS

- Existen 4 tipos básicos de eventos:

-SOBRE EL FOTOGRAMA
-SOBRE EL RATÓN
-SOBRE TECLA
-SOBRE CLIP

- FOTOGRAMA

Permiten que se ejecute los script definidos en ellos. Por lo tanto su radio de acción es instante del que consta un frame y repercute en todos los elementos que están en ese frame.

trace(“hola mundo”); ⟶ **Hola mundo**

Salida

AS

EVENTOS

- FOTOGRAMA

Permiten que se ejecute los script definidos en ellos. Por lo tanto su radio de acción es instante del que consta un frame y repercute en todos los elementos que están en ese frame.

trace(“hola mundo”); ⟶ **Hola mundo**

- RATÓN

Permiten aplicar acciones a un evento del ratón en un botón por el usuario.

```
on(evento){

sentencia ó acción;

}
```

Ej:

```
on(press){

trace(“hola mundo ”);

}
```

eventos:

- press
- release
- releaseOutside
- rollOver
- rollOut
- dragOver
- dragOut

- FOTOGRAMA + RATON (HANDLERS)

También podemos realizar la acción de un evento del ratón sobre un botón desde el fotograma con esta estructura

```
onRelease = function() {
// acción
}
```

Ejemplo:

Nombre de instancia
Del botón

```
hola_bot.onRelease = function(){
        trace("hola");

        }
```

AS

EVENTOS

- TECLADO

Permiten aplicar acciones a un evento del ratón por el usuario.

```
on(keyPress “tecla”){

sentencia ó acción;

}
```

Ej:

```
on(keyPress “<Enter>”){

trace(“hola mundo ”);

}
```

eventos:

- Left
- Right
- Home
- End
- Insert
- Delete
- Backspace
- Enter
- Up
- Down
- PageUp
- PageDown
- Tab
- Escape
- Scape
- carateres (“a”)

AS

EVENTOS

- CLIP

Eventos que disparan acción en relación con los Movieclip

```
onClipEvent(evento){

sentencia ó acción;

}
```

Ej:

```
onClipEvent(enterFrame){

trace(“hola mundo ”);

}
```

eventos:

- load
- unload
- enterFrame
- mouseDown
- mouseUp
- mouseMove
- KeyDown
- keyUp
- data

- CLIP- AVANZADO – (para Flash MX)

Eventos que disparan acción en relación con el Movieclip con una sintaxis distinta, más rápida y directa en ejecución. Podemos trabajar con ellos desde el frame, no es necesario estar dentro del moviclip.

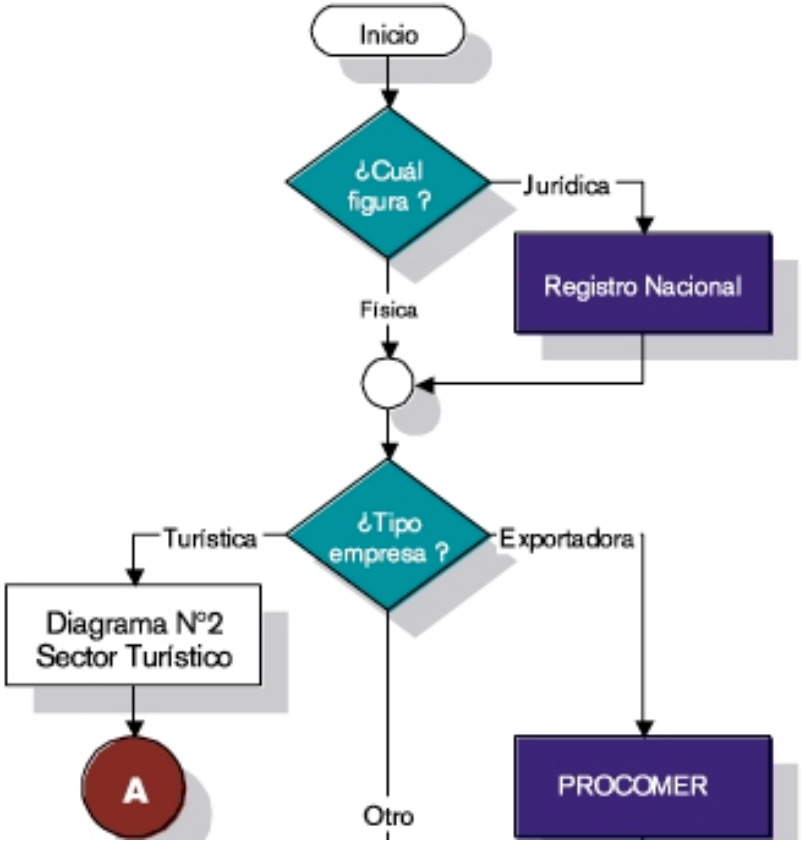
```
instanciaClip.evento = fuction(){

sentencia ó acción;

}
```

```
cuBo.onEnterFrame = function(){

trace("hola mundo ");

}
```

AS

ESTRUCTURAS

Las estructuras de condición e interacción nos permiten programar la aplicación para que se comporte de una manera ó de otra dependiendo de la situación.



- **IF**
- **WITH**
- **FOR**
- **WHILE**
- **BREAK**
- **CONTINUE**

AS

SELECCIÓN ESTRUCTURAS

if

Sentencia que evalúa expresiones lógicas. Expresa condición Para la realización de la instrucción. Si es afirmativa la respuesta ejecuta una, Si no lo es ejecuta la otra

```
if(condición){

accion;

}
```

```
if(cuBo >= 5){

Trace(“hola mundo”);

}
```

```
if(condición){

accion1;

}else{
accion2;
}
```

```
if(cuBo >= 5){

trace(“hola mundo”);

}else{
trace(“no hay hola mundo”);
}
```

while

Sentencia que **ejecuta las acciones tantas veces** sin parar hasta el momento en que no se cumpla la condición.

```
while(condición){
accion1;
accion2;
Etc….
}
```

```
while(i<=10){
trace(“hola mundo”);
}
```

for

Sentencia que **ejecuta las acciones tantas veces** como determine la condición

for(valor inicial de la variable; condición de continuidad; modificación de la variable**){**
accion1;
accion2;
Etc….
}

```
for(i=1; i<=10; i++){
trace(“hola mundo”);
}
```

break

Interrupción del bucle. Produce la detención de la actividad de las sentencias que se ejecutan dentro del bucle

```
i=0;
while (i<=5){

i++;

if (i==3) break;

trace(“hola mundo”);

} // fin while
```

continue

NO se ejecutan las acciones que hay por debajo de ella.
Por lo que en este caso, ejecutará las acciones cuando el valor de i es 1,2,4,5. No las ejecuta cuando es 3.

```
i=0;
while (i<=5){

i++;

if (i==3) {
continue;
} // fin if

trace(“hola mundo”);

} // fin while
```

switch

Prueba una condición que ejecuta sentencias si la condición devuelve el **valor true.**

```
switch (numberito) {
case 1:
trace ("1 a sido elegido");
break; // es importante para romper cuando un caso ha sido elegido
case 2:
trace (“2 a sido elegido“);
break;
case 3:
trace (“3 a sido elegido“);
break;
default: // si ningún case coincide se activa default
trace (“niguno ha sido elegido” )
}
```

function

Son trozos, fragmentos de código recogidos en un paquete al cual más tarde podemos llamarlo con un simple nombre.

MODO 1

```
function nombre de la función ( parámetros){
accion1;
accion2;
Etc….
}
```

MODO 2

```
nombre de la función = function ( parámetros){
accion1;
accion2;
Etc….
}
```

Si no tiene parámetros dejamos vacíos los paréntesis

AS

FUNCIONES ESTRUCTURAS

function

Son trozos, fragmentos de código recogidos en un paquete al cual más tarde podemos llamarlo con un simple nombre.

Situada en el frame

```
function sumaMe (){
coSita = 5+6;
trace(cosita);
}
```

Llamada desde un botón por ejemplo:

```
on(release){

sumaMe();

}
```

:)

Prof. Moisés Mañas Carbonell
Dpto. Escultura. UPV
moimacar@esc.upv.es