ACTIONSCRIPT (AS)

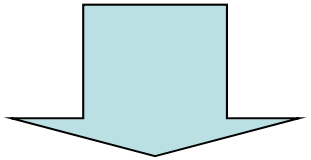
Dpto. Escultura. UPV.

[objetos]

OBJETOS

Un objeto es elemento que interviene en nuestra película, el cual contiene una serie de características que lo definen, llamados **atributos** y una serie de actividades por defecto llamadas **métodos.**

Este tipo de elementos pertenecen a los tipos de lenguajes **OOP** (**Objetcs Oriented Programming**)



El objeto principal o "**padre"** de donde proceden, se denomina **clase.(Este reúne todas las características posible de dicho objeto y definirá las propiedades y métodos para los posteriores "hijos".**

ACTION(AS)

OBJETOS

OBJETOS DE LA CLASE **COCHE**

Obj. deportivo

Obj. F1

Obj. T/T

CLASE **COCHE:**
métodos- arrancar, ir, girar, parar
Atributos – color, velocidad, carburante

La clase define un miembro real o una entidad abstracta. Define por lo tanto el comportamiento y atributos de un grupo de objetos de características similares.

OBJETOS

Construcción de un objeto:

```
Coche = new Object();
Coche.marca="seat";
Coche.puertas=5;
Coche.velocidad=200;
Coche.ruedas=4;
Coche.combustible="diesel";

// salida en la consola de salida de Flash

trace("marca_> "+Coche.marca);
trace("puertas_> "+Coche.puertas);
trace("velocidad_>"+Coche.velocidad);

// salida en variable en el campo de texto

marca_var = Coche.marca;
puertas_var = Coche.puertas;
Velocidad_var = Coche.velocidad;
```

OBJETOS EN FLASH

(No están todos pero una selección útil para el curso)

OBJETOS

ARRAY

Ver clase anterior para ver características. (No están todos pero sí algunos útiles para el curso)

Métodos:

nom_array.**length** Su valor es el número de elementos que tiene

```
Numeritos =[1,2,3,4,5,6];
Numeritos.length;
```
→ Nos dará 6

nom_array.**shift** Su resultado será el valor de la primera posición

```
Numeritos =[1,2,3,4,5,6];
Numeritos.shift;
```
→ Nos dará 1

nom_array.**pop** Su resultado será el valor la última posición

```
Numeritos =[1,2,3,4,5,6];
Numeritos.pop;
```
→ Nos dará 6

OBJETOS

ARRAY

Métodos:

nom_array.**sort** Ordena los valores de menor a mayor

```
Numeritos =[6,5,3,4,2,1];
Numeritos.sort;
```
→ Organizará 1,2,3,4,5,6

nom_array.**reverse** Ordena los valores en orden inverso

```
Numeritos =[1,2,3,4,5,6];
Numeritos.reverse;
```
→ Nos dará 6,5,4,3,2,1

nom_array.**push** Añade valores al final, pasándole un parámetro

```
Numeritos =[1,2,3,4,5,6];
Numeritos.push(7,8);
```
→ Numeritos=[1,2,3,4,5,6,7,8];

Ej:

```
Numeros=[1,3,4,5];
trace(Numeros.length);
Numeros.push(6,7);
trace(Numeros.length);
```
→ Dará 4 y 6

OBJETOS

DATE

Permite jugar con la hora y fecha del sistema

Constructor:

Mifecha = new Date();

Métodos:

nom_date.**getdate**	Devuelve el día del més

Hoy= Mifecha.getDate(); → Nos dará número que es el día de hoy

nom_date.**getDay**	Devuelve el dia de la semana en formato numérico 0 es Domingo y 6 Sábado

Hoy= Mifecha.getDay(); → Nos dará el número del día

OBJETOS

DATE

Métodos:

nom_date.**getMonth** Devuelve el més (0=Enero, 11=Diciembre)

Minutos= Mifecha.getMonth(); ⟶ 3 (Abril)

nom_date.**FullYear** Devuelve el año actual

Anyo= Mifecha.FullYear(); ⟶ 2005

nom_date.**getHours** Devuelve la Hora

Hora= Mifecha.getHours(); ⟶ 14

nom_date.**getMinutes** Devuelve los minutos

Minutos= Mifecha.getMinutes(); ⟶ 25

nom_date.**getSeconds** Devuelve los segundos

Segundos= Mifecha.getSeconds(); ⟶ 45

OBJETOS

DATE

Métodos:

De la misma manera que obtenemos valores con el **GET** podemos Pasarle los valores al objetos con el **SET**. **Se utilizan los mismos métodos**

Ej:

Midia_ideal = new Date();
Midia_ideal.**setdate(14);**

Mi día ideal será 14 independientemente del reloj del sistema

OBJETOS

NUMBER

Permite manipular valores numéricos

Constructor:

numerito = new Number();

Métodos:

numerito.**toString**	Convertirá el número en un valor de tipo string

Ej:
numerito = new Number(1234);
numerito.toString(); → "1234"

OBJETOS

STRING

Permite manipular valores de cadenas

Constructor:

palabrita = new String(“hola esto es una cadena”);

Métodos:

palabrita.**charAt**

Nos devuelve el carácter de la posición indicada. Desde 0 hasta **lenght-1**

palabrita.charAt(0); → “h”

palabrita.charAt(palabrita.lenght-1); → “a”

palabrita.**slice**

Devuelve la cadena contenida entre los parámetros que le apliquemos

palabrita.slice(0,6); → “hola es”

OBJETOS

STRING

Métodos:

palabrita.**split**

Separa las cadenas por medio del parámetro pasado

palabrita.split(“a”); → “hol. esto es un. c.den.”

palabrita.**substr**

Devuelve una cadena de tanto valores como indique el segundo parámetro desde la posición que indique el primer parámetro.

palabrita.substr(3,9); → “a esto es”

palabrita.**substring**

Devuelve la cadena contenida entre los parámetros que le apliquemos como principio y final de captura

palabrita.substring(0,11); → “hola esto es”

OBJETOS

STRING

Métodos:

palabrita.**toLowerCase** Convierte todo a minúsculas

```
minusculas = palabrita.toLowerCase();
```

palabrita.**toUpperCase** Convierte todo a MAYUSCULAS

```
minusculas = palabrita.toUpperCase();
```

OBJETOS

MOVIECLIP

Muchos de estos funcionan igual que las acciones que se ejecutan sobre eventos de un moviclip.

Constructor:

Instancia_movieclip.método;

Métodos:

instancia.**getBytesLoaded** — Devuelve el número de Bytes cargados del clip

```
datos = bola.getBytesLoaded();
trace(datos);
```

bola.**getBytesTotal** — Nos devuelve el número de Bytes total del clip

```
datos = bola.getBytesTotal();
trace(datos);
```

OBJETOS

MOVIECLIP

Métodos:

instancia.**getBytesLoaded** Devuelve el número de Bytes cargados del clip

```
datos = bola.getBytesLoaded();
trace(datos);
```

instancia.**getBytesTotal** Nos devuelve el número de Bytes total del clip

```
datos = bola.getBytesTotal();
trace(datos);
```

instancia.**getURL** Nos abre una url, con la variable de forma que necesitamos

```
bola.getURL("http://www.upv.es",_black);
```

OBJETOS

MOVIECLIP

Métodos:

instancia.**hitTest**

Comprueba si dos movieclips colisionan

bola2.hitTest(bola)

instancia.**gotoAndPlay**

Sirve para ir a un fotograma ó etiqueta de un determinado movieclip y seguir.

Bola.gotoAndPlay(5);

instancia.**gotoAndStop**

Sirve para ir a un fotograma ó etiqueta de un determinado un movieclip, y parar.

Bola.gotoAndStop(5);

instancia.**play**

Sirve para reproducir el movieclip.

Bola.play();

OBJETOS

MOVIECLIP

Métodos:

instancia.**nextFrame**

Sirve para ir al siguiente fotograma de un movieclip.

```
Bola.nextFrame();
```

instancia.**prevFrame**

Sirve para ir al fotograma previo de un movieclip.

```
Bola.prevFrame(5);
```

instancia.**stop**

Para la animación de los fotogramas de un movieclip, y parar.

```
Bola.stop();
```

instancia.**currentFrame**

Sólo es de lectura, es un **propiedad** nos dice en que frame se encuentra el reproductor de la animación

```
Bola.gotoAndStop(_currentframe + 5);_
```

OBJETOS

MOVIECLIP

Métodos:

instancia.**loadMovie**

Carga una película .swf o un .jpeg dentro del clip

```
Bola.loadMovie(“imagen_bola.swf”);
```

instancia.**loadVariables**

Utilizado para cargar variables que se encuentran en una URL o TXT dentro de un Movieclip

```
loadVariables ("archivo.txt", "_root.clip");
```

instancia.**loadVariablesNum**

Utilizado para cargar variables que se encuentran en un fichero TXT en un nivel específico (0).

```
loadVariablesNum ("archivo.txt", 0);_
```

instancia.**StartDrag**

Hace arrastrable el movieclip.

```
Bola.startDrag();
```

instancia.**StopDrag**

Desactiva la cualidad arrastrable el movieclip.

```
Bola.StopDrag();
```

OBJETOS

MOVIECLIP

Propiedades:

instancia.**enabled**

Indica si los movieclip que tienen la propiedad debotón estan activados ó no (da un booleano true/false)

```
if(boton.enabled= true){
        trace ("si");
}
```

instancia.**_frameloaded**

Nos devuelve el número de fotogramas que han sido cargados de un moviclip

instancia.**_totalframes**

Nos devuelve el número total de fotogramas que contiene un movieclip

```
if (_framesloaded >= _totalframes) {
gotoAndPlay (5);
} else {
gotoAndPlay (1);
}
```

OBJETOS

MOVIECLIP

Propiedades:

instancia._**alpha**	Propiedad de hacerse transparente el movieclip

```
Bola._alpha =50;
```

instancia._**height**	Propiedad que nos actúa sobre el alto del movieclip

```
Bola._height =250;
```

instancia._**width**	Propiedad que nos actúa sobre el ancho del movieclip

```
Bola._width =250;
```

instancia._**rotation**	Propiedad que contiene la rotación de un movieclip

```
Bola._rotation =45;
```

OBJETOS

MOVIECLIP

Propiedades:

instancia._**visible**

Propiedad de hacerse visible ó invisible el movieclip

```
bola._visible = 0;
```

instancia._**x**

Propiedad que contiene la posición horizontal del movieclip en la pantalla

```
Bola._x =250;
```

instancia._**y**

Propiedad que contiene la posición vertical del movieclip en la pantalla

```
Bola._y =250;
```

instancia._**xscale**

Propiedad que especifica la propiedad de escalar un movieclip en horizontal.

```
Bola._xscale = 50;
```

instancia._**yscale**

Propiedad que especifica la propiedad de escalar un movieclip en vertical

```
Bola._yscale = 20;
```

ACTION(AS)

OBJETOS

MOVIECLIP

Propiedades:

instancia.**_xmouse**

Contiene la información de la coordenada horizontal del ratón respecto a un clip

```
DondeX =bola._xmouse;
```

instancia.**_ymouse**

Contiene la información de la coordenada vertical del ratón respecto a un clip

```
DondeY=bola._ymouse;
```

instancia.**_useHandCursor**

Especifica cambio del cursor al icono de mano sobre el movieclip.

```
Bola._useHandCursor=true;
```

MISCELANIA MOUSE

Mouse.Hide();

Esconde el ratón

Mouse.Show();

Muestra el ratón

OBJETOS

CONTROL MOVIECLIPS

instancia.**duplicateMovieClip** Método que duplica un movieclip

Bola.**duplicateMovieClip**(“nueva_bola”,2);

Movieclip a duplicar_

Nombre del nuevo MC

Nivel Donde se situará

OBJETOS

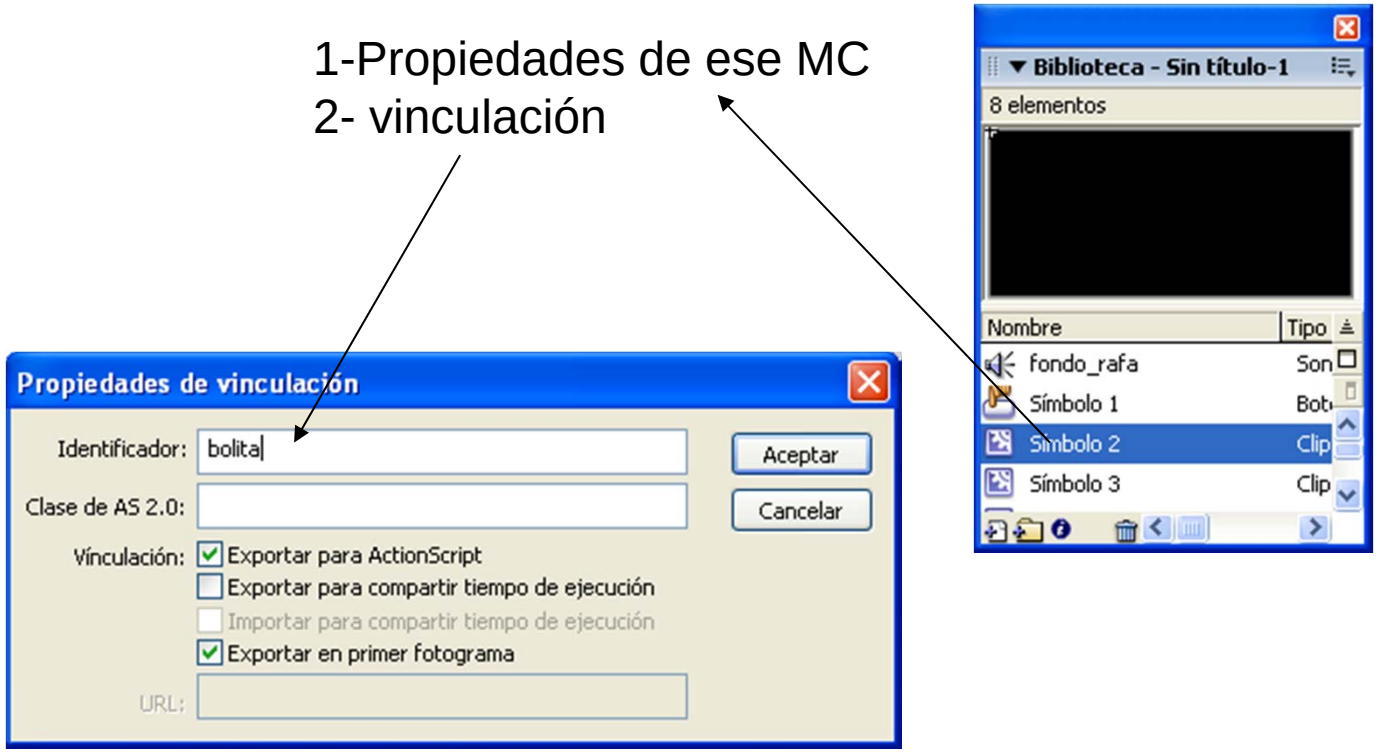
CONTROL MOVIECLIPS

instancia.**attachMovie**

attachMovie(“bolita”,”bola”, 2);

Nos sirve para cargar movieclips que se encuentran en la biblioteca de la película.

El MC que vamos a cargar deben de tener un nombre de identificador de vinculación



OBJETOS

CONTROL MOVIECLIPS

instancia.**attachMovie**

Nos sirve para cargar movieclips que se encuentran en la biblioteca de la película.

Los que vamos a cargar deben de tener un nombre de vinculación

attachMovie(“bolita”,”bola”, 2);

- nombre vinculado del MC que vamos a insertar
- MC donde se deposita
- Nivel donde se situará

- Ejemplo: Carga y posicionamiento en el escenario

```
attachMovie(“bolita”,”bola”, 2);
bola._x = 200;
bola._y = 15;
```

OBJETOS

CONTROL MOVIECLIPS

instancia.**createEmptyMovieclip()**

Crea MC vacíos para ser usados, por ejemplo para dibujar en ellos ó utilizar un loadMovie();

_root.createEmptyMovieClip(“vacio”,2);

Nombre del MC vacio

Nivel donde se situará

ACTION(AS)

OBJETOS

CONTROL MOVIECLIPS

.getProperty(inst_mc, propiedad)

Muy útil. Obtiene la propiedad específica de un MC. En este caso donde esta en X bola.

Donde = getProperty(_root.bola,_x);

.setProperty(inst_mc, propiedad, valor)

setProperty(“bola", _alpha, "30");_

Muy útil. Envia la propiedad específica de un MC. En este caso donde esta en X bola.

OBJETOS

CONTROL MOVIECLIPS

Sentencia:

```
tellTarget(“ruta_Movieclip)“{
acción();
 }_
```

La sentencia tellTarget controla la instancia de clip de película

Cadena que especifica la ruta de destino de la línea de tiempo que debe controlarse

```
tellTarget(“_root.bola“){

 gotoAndPlay(2);

}_
```

Es igual que = _root.bola.gotoAndPlay(2);

OBJETOS

Key

Key.getCode();

Devuelve el valor de código de la última tecla presionada.

Key.getAscii();

Devuelve el código ASCII de la última tecla presionada o soltada

Key.isDown();

Detecta la tecla que esta presionada y dentro de los parentesis colocaremos el código de la tecla resultante de Key.getCode();

```
if(Key.isDown(32)){
        trace("hola");
        };
```

Key.BACKSPACE

Detecta una tecla específica: BACKSPACE, CONTROL, ENTER, UP, DOWN, ETC…..

OBJETOS

listener

adListener();

Detector de eventos por ejemplo mover el ratón, una tecla, cambiar la pantalla.

```
// crear un LISTENER , creamos un objeto y lo llamamos “midetector”
var midetector:Object = new Object();
// hay que adquirir una clase al listener en este caso la Key.
Key.addListener(midetector);
// para detectar el evento del ratón
//Mouse.addListener(midetector);
midetector.onKeyDown = function(){
            trace("has pulsado una tecla");
            }
// limpiar el listener
Key.removeListener(midetector);
```

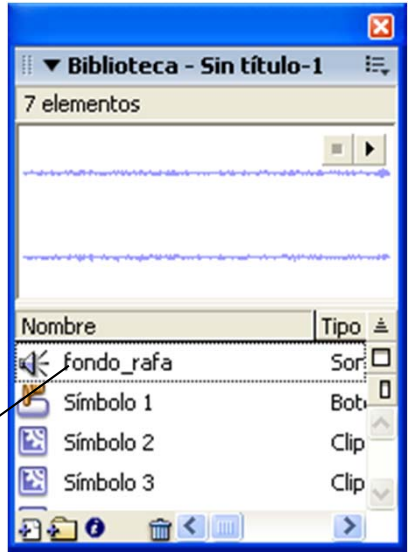
removeListener();

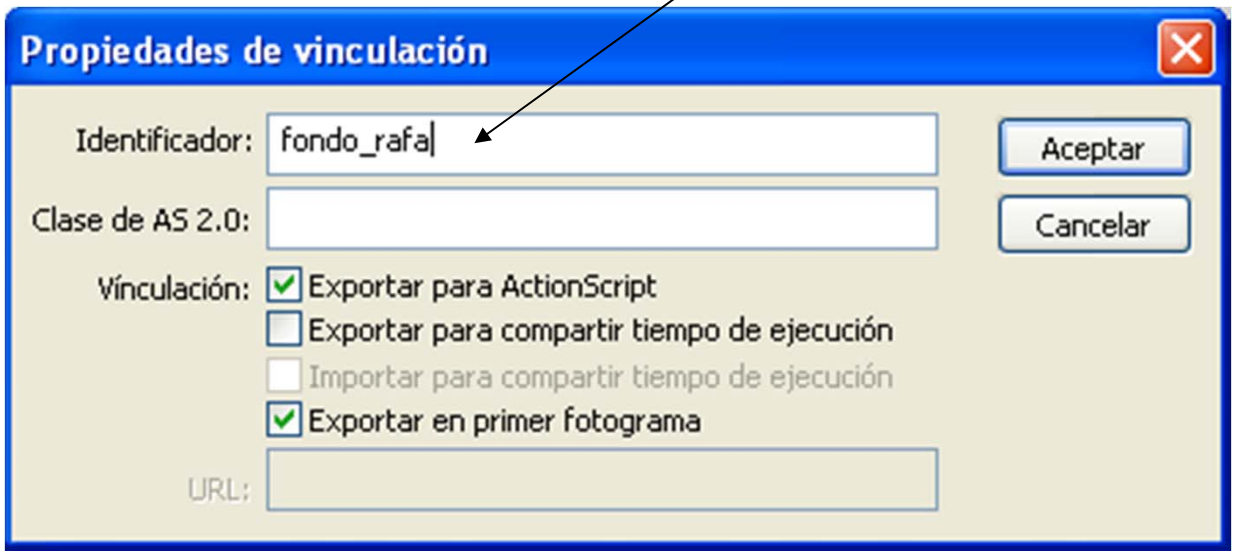
Limpia el detector de evento

OBJETOS

SOUND

1- importar el sonido a la biblioteca.
2- seleccionamos vinculación(botón derecho)
3- le asignamos un identificador (DNI)
4 – Activamos [exportar para Action + exporta en primer fotograma].





OBJETOS

SOUND

Este objeto nos sirve para utilizar sonido en flash, y tener un control casi total sobre él.

Constructor:

```
Sonido = new Sound(inst_movieclip);
Sonido = new Sound(_root);
Sonido = new Sound(this);
```

-Si es _**root** la instancia sonido esta en la película principal

-Si es el nombre de **inst_movieclip** pertenece a ese Movieclip

- Si es **this** sonido pertenece a la linea de tiempo en la que escribe el código.

OBJETOS

SOUND

Métodos:

Sonido.**attachSound**

Llama al sonido de la biblioteca (nombre=identificador)

```
Sonido.attachSound("rock");
```

Sonido.**loadSound**

Carga el sonido desde un directorio externo. El parámetro booleano (true/false) significa: true = hace play mientras carga el sonido. False= se espera hacer play cuando carga todo el sonido

```
Sonido.loadSound("nombresonido.mp3",true);
```

Sonido.**setVolume**

Le manda el volumen del sonido (entre 0 y 10)

```
Sonido.setVolume(10);
```

Sonido.**start**

Comienza el sonido (PLAY)

```
Sonido.start();
Sonido.start(0,3);
```

→ 0= Donde comienza a reproducir

999 = cuantas veces se repite (loop)

Sonido.**stop**

Para el sonido

```
Sonido.stop();
```

OBJETOS

SOUND

Métodos:

Sonido.**setPan**

Panoramiza el sonido a la derecha (100)y a la izquierda(-100)

```
Sonido.setPan(-100);
```

Sonido.**getBytesTotal**

Devuelve los bytes totales del sonido (de lectura)

```
Sonido.getBytesTotal
```

Sonido.**getBytesLoaded**

Devuelve los Bytes que se han cargado del sonido(de lectura)

```
Sonido.getBytesLoaded
```

Sonido.**duration**

Devuelve la duración del sonido en milisegundos

```
Sonido.duration
```

ACTION(AS)

OBJETOS

MICROPHONE

System.showSettings(2);

Muestra la ventana de valores de settings
0 = Privacidad
1 = Almacenamiento total
2 = microfono
3 = camara

var micro:Microphone = new Microphone();

Constructor de un objeto microphone

micro**.get();**

Captura el micrófono

micro = Microphone.get();

attachAudio(nombre del microfono);

Especifica la fuente de audio que debe reproducirse y donde _**root.attachAudio(false)**;con "false" no se reproduce el audio.

_root.attachAudio(micro);

objetomicro.**setGain(**valor entre 0 y 100);

Valor(entre 0-100) con el que el micrófono debe reforzar la señal(Ganacia). Valor predeterminado es 50

OBJETOS

MICROPHONE

objetomicro.**rate = valornumérico;**

Frecuencia a la que está capturando el sonido el micrófono, expresada en kHz(5, 8, 11, 22, 44)

```
micro.rate = 22;
```

objetomicro.**name;**

Cadena que muestra el dispositivo del sistema que usa el micro

```
micro.name;
```

objetomicro.**setUseEchoSuppression(true);**

Supresión de eco del micro

```
micro.setUseEchoSuppression(true);
```

objetomicro.**activityLevel**

activity level detecta la actividad del micro entre 0 y 100

```
trace(micro.activityLevel);
if (micro.activityLevel>=50) {
    trace("mas de 50");
}
```

ACTION(AS)

OBJETOS

CAMERA

System.showSettings(2);

Muestra la ventana de valores de settings
0 = Privacidad
1 = Almacenamiento total
2 = microfono
3 = camara

miCamara = Camera.get(); Constructor de un objeto CAMERA

miVideo.attachVideo (miCamara); adjuntamos el objeto video de la biblioteca

miVideo.smoothing=3;

:)

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