Lección 10: Conceptos Básicos de Programación

En esta Lección aprenderás algunos conceptos básicos de programación en AutoPlay. Aunque puedes llevar a cabo muchas tareas en AutoPlay sin ningún conocimiento de programación, aunque sea un poquito de práctica en programación puede hacer una gran diferencia. Puedes lograr mucho más en un proyecto con un poco de programación que lo que harías sin ella. La programación abre la puerta a todo tipo de técnicas avanzadas, desde acciones que son ejecutadas sólo cuando se cumplen condiciones específicas, hasta funciones que puedes definir, nombrar y luego llamar desde cualquier otro lugar.

¿Qué aprenderás?

En esta lección aprenderás cómo:

- Mostrar un mensaje
- Utilizar una variable
- Añadir una declaración *if*
- Probar valores numéricos
- Ajustar el texto de un objeto de Botón
- Eslabonar (unir) cadenas
- Comparar cadenas
- Utilizar un ciclo *for*
- Crear funciones

¿Cuánto tiempo tomará esto?

Esta lección dura aproximadamente 30 minutos.

Comenzando la Lección

Para esta Lección, vamos a crear un nuevo proyecto desde cero. Si estás continuando desde la Lección 9, deberías tener ejecutándose el AutoPlay con el proyecto *Tutorial* aún abierto.

Si no es así, necesitarás abrir primero el AutoPlay Media Studio.

1) Si tienes ejecutándose el Autoplay con el proyecto Tutorial abierto, elige *File > New*.

Al elegir *File* > *New* se abrirá el cuadro de diálogo *New Project Options*.

Una vez abierto, avanza al paso 3.

2) De otro modo, si no tienes ejecutándose el AutoPlay, utiliza el menú *Inicio* de Windows para abrir el programa AutoPlay Media Studio. Cuando el cuadro de diálogo de Bienvenida aparezca, haz click en "*Create a new project*".

Encontrarás el AutoPlay Media Studio en:

Inicio > *Programas* > *Indigo Rose Corporation* > *AutoPlay Media Studio 7.0*

Una vez que el programa inicie y el cuadro de diálogo de Bienvenida aparezca, haz click en la opción "*Create a new project*". Al hacer esto se abrirá el cuadro de diálogo *New Project Options*.

3) Cambia el nombre del proyecto a *Lección 10*.

Queremos reemplazar el texto por defecto por el de un nombre único para este proyecto, así que selecciona todo el texto del campo *Project Name* y escribe *Lección 10*.

Este nombre será utilizado por la carpeta del proyecto y el archivo de proyecto.

4) Selecciona *Blank Project* y haz click en *Create Project Now*

Cuando haces click en *Create Project Now*, el cuadro de diálogo *New Project Options* se cierra, AutoPlay configura la carpeta del proyecto y el archivo de proyecto L*ección 10* y un nuevo proyecto en blanco es cargado dentro del ambiente de diseño.

Ahora estás listo para aprender algunos conceptos básicos de programación.

Mostrar un Mensaje

Bueno, primero lo primero. Antes de diseñar algo sofisticado, haremos un script que realice algo muy simple, como mostrarle un mensaje al usuario.

Sin embargo, antes que nada necesitamos un lugar en el cual colocar nuestro script. Por conveniencia, utilizaremos el evento *On Click* de un objeto de Botón.

1) Agrega un objeto de Botón en la página

Una forma rápida de añadir un objeto de Botón es hacer click derecho en la página y elegir *Button* desde el menú contextual. Luego simplemente selecciona el botón que quieras utilizar desde la lista de archivos de botón en el cuadro de diálogo *Select File* y haz click en *OK*.

Nota: No importa cuál botón elijas, simplemente selecciona cualquiera que te agrade.

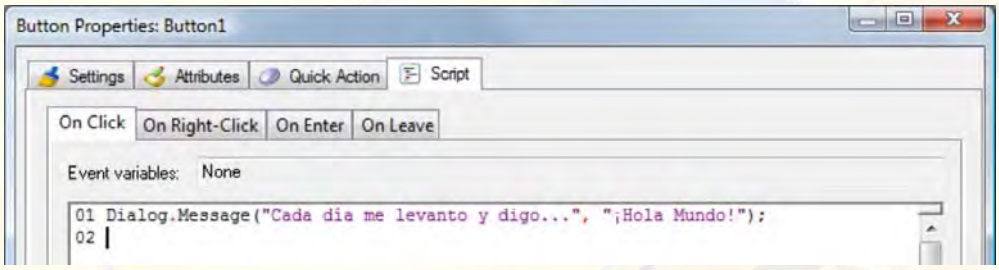
2) Haz doble click en el objeto de Botón, luego haz click en la pestaña *Script* y por último haz click en la pestaña del evento *On Click*.

Queremos añadirle una acción al evento *On Click* del objeto, así que abre el cuadro de diálogo *Button Properties* (Propiedades del Botón), conmuta a la pestaña *Script* y asegúrate que estás editando el Script para el evento *On Click.*

3) En la primera línea escribe:

Dialog.Message("Cada día me levanto y digo...", "¡Hola Mundo!");

Este script consta de una única acción *Dialog.Message* la cual asigna "*Cada día me levanto y digo...*" como la cadena que se va a mostrar en la barra de título, y "*¡Hola Mundo!*" como la cadena que mostrará el cuadro de diálogo. Cuando el script esté terminado, deberá mostrar algo como esto:



Nota: Aunque este script tiene una sola acción, los scripts pueden ser mucho más extensos. De hecho, no hay límite para el tamaño de los scripts.

4) Haz click en *OK* para cerrar el cuadro de diálogo *Button Properties*. Presiona *F5* para previsualizar el proyecto. Cuando aparezca la vista previa, haz click en el botón para ver el mensaje.

Al hacer click en el botón, se activa su evento *On Click* y se ejecuta cualquier script que haya sido añadido en él. En este caso, el script consta de una única llamada a la acción *Dialog.Message*, la cual despliega el siguiente mensaje:

5) Haz click en *Aceptar* para cerrar el mensaje del cuadro de diálogo y sal de la vista previa.

Haz click *Aceptar* para cerrar el mensaje, y luego sal de la vista previa (haciendo click en el icono "X") para regresar al ambiente de diseño.

Consejo: También puedes salir de la vista previa presionando *Alt+F4.*

Utilizando una Variable

Una de las más poderosas características de la programación es la habilidad para hacer uso de *variables*. Esencialmente las variables son sólo "*apodos*" o "*contenedores*" para valores que pueden necesitar ser modificados o reutilizados en el futuro. A cada variable se le da un nombre, el cual puedes utilizar para acceder a su valor actual en tu script.

Nota: Decimos que los valores son "*asignados a*" o "*almacenados en*" las variables. Si describes una variable como un recipiente que puede contener un valor, asignar un valor a una variable es como "*colocar*" ese valor dentro del recipiente.

Colocas un valor dentro de una variable al asignarle el valor con un signo de igual (=). Por ejemplo, el siguiente script le asigna el valor de 10 a una variable llamada "cantidad".

```
cantidad = 10;
```

Puedes cambiar este valor cuando quieras al asignarle un valor diferente a la variable. (El nuevo valor simplemente reemplaza al viejo). Por ejemplo, el siguiente script le asigna el valor de 45 a la variable cantidad, reemplazando al número 10:

```
cantidad = 45;
```

...y el siguiente script le asigna la cadena "¡Woohoo!" a la variable, reemplazando al número 45:

```
amount = "Woohoo!";
```

Date cuenta que puedes reemplazar fácilmente un valor numérico por una cadena en una variable. Tener un número o una cadena en una variable no la "bloquea" para que sólo acepte ese tipo de valor. A las variables no les importa el tipo de datos que contengan.

Esta habilidad para contener información cambiable es lo que hace a las variables tan útiles.

1) Haz doble click en el objeto de Botón. En el script *On Click*, reemplaza la cadena "*¡Hola Mundo!*" por una variable nombrada *strMsg*.

Sólo edita el script del evento *On Click* del objeto de Botón de modo que en su lugar se vea como éste:

```
Dialog.Message("Cada día me levanto y digo...", strMsg);
```

Esto hará que cuando sea ejecutada la acción *Dialog.Message*, muestre el valor actual de la variable *strMsg*. Sin embargo, antes de que probemos esto, necesitamos asignarle un valor a la variable en algún lugar.

Nota: Una práctica común entre los programadores es darle a sus variables nombres con prefijos que les ayuden a recordar lo que las variables se supone que contienen. Uno de esos prefijos es "str", el cual es utilizado para indicar que una variable contiene una string (cadena). Otro prefijo común es "n" para un valor numérico (por ejemplo nCuenta, nTotal) y "b" para un valor Booleano true/false (por ejemplo bCargado, bListoParaIniciar).

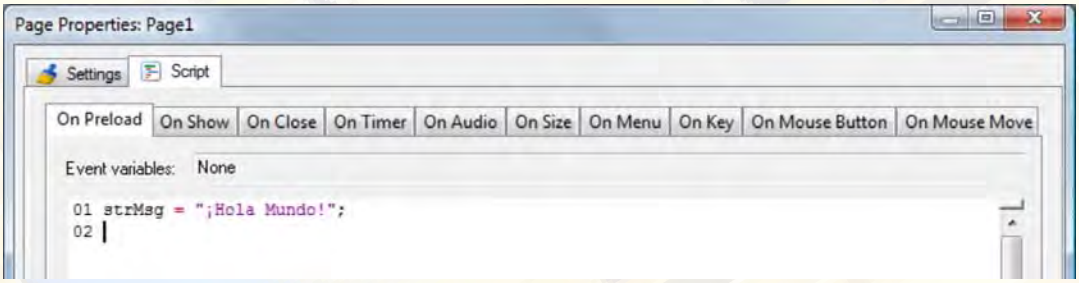
2) Haz click en *Aceptar* para cerrar el cuadro de diálogo Propiedades.

Cuando hayas modificado el script *On Click*, haz click en *Aceptar* para aceptar los cambios y cerrar el cuadro de diálogo Propiedades

3) Haz doble click sobre la página y luego haz click en la pestaña *Script*. En la primera línea del script *On Preload*, escribe lo siguiente:

strMsg = "¡Hola Mundo!";

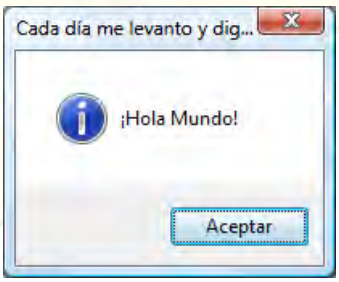
El script deberá verse como éste cuando esté terminado:



Esto le asignará la cadena *"¡Hola Mundo¡"* a la variable nombrada *strMsg* durante el evento *On Preload* de la página. El evento *On Preload* es activado tan pronto como la página ha sido creada en la memoria, justo antes de que ésta aparezca en la pantalla. Esto la hace que sea un buen lugar para colocar cualquier script de inicialización, tal como establecer valores por defecto o preparar variables que serán utilizadas una vez que la página sea mostrada.

4) Presiona *F5* para previsualizar el proyecto. Cuando aparezca la vista previa, haz click sobre el botón para ver el mensaje.

La acción *Dialog.Message* muestra el cuadro de mensaje de diálogo, igual que antes.



Observa que el nombre de la variable *strMsg* no se encuentra en ninguna parte… en lugar de ésta, se muestra el valor que está actualmente en la variable. En este caso, es el valor que le fue asignado a la variable en el evento *On Preload* de la página.

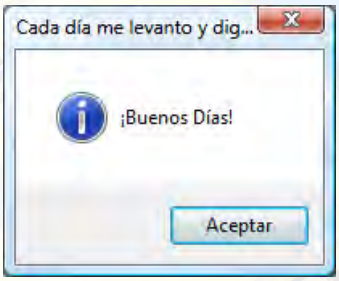
5) Sal de la vista previa. En el script *On Preload* de la página, cambia el valor que está asignado a la variable *strMsg* por el de *"¡Buenos Días!"*.

Haz doble click en la página y edita el script del evento *On Preload* de modo que en su lugar se vea como éste:

strMsg = "¡Buenos Días!";

6) Presiona *F5* para previsualizar el proyecto. Cuando se abra la vista previa haz click sobre el botón para ver el mensaje.

Esta vez, el mensaje se verá como éste:



Como puedes ver, acabas de cambiar el mensaje sin ni siquiera tocar la acción *Dialog.Message*.

Ahora imagina si tuvieras tales acciones *Dialog.Message* en cincuenta páginas a lo largo de tu proyecto, y decidieras que quieres cambiarles el texto. Con las variables y un poco de planeación, tales cambios son pan comido.

Añadiendo una Declaración *If*

La declaración *if* les proporciona a tus scripts la habilidad de tomar decisiones, y hace una cosa u otra dependiendo de las circunstancias.

Cada declaración *if* consta de una *condición* (o "prueba"), seguida por un paquete de acciones de script que sólo será ejecutado si la condición resulta ser verdadera.

La sintaxis básica es:

if *condición* then
    hacer algo aquí
end

Por ejemplo:

```
if edad < 18 then
   Dialog.Message("¡Disculpa!", "Debes tener 18 años o más para acceder a este CD.");
   Application.Exit();
end
```

El script de arriba verifica si el valor de una variable nombrada "*edad*" es menor que 18. Si lo es, entonces coloca un mensaje diciendo *"Debes tener 18 años o más para acceder a este CD."*, y luego cierra inmediatamente la aplicación.

Por ejemplo, si primero ajustamos edad a 17:

```
edad = 17;
if edad < 18 then
   Dialog.Message("¡Disculpa!", "Debes tener 18 años o más para acceder a este CD.");
   Application.Exit();
end
```

...el bloque de script que está entre las palabras clave "*then*" y "*end*" será ejecutado debido a que *17* es menor que *18*. En este caso, decimos que la condición de la declaración *if* ha sido "*aprobada*".

Sin embargo, si ajustamos edad a 20:

```
edad = 20;
if edad < 18 then
   Dialog.Message("¡Disculpa!", "Debes tener 18 años o más para acceder a este CD.");
   Application.Exit();
end
```

... el bloque de script entre las palabras clave "*then*" y "*end*" no será ejecutado debido a que *20* no es menor que *18*. Esta vez decimos que la condición de la declaración *if* ha "*fallado*"

Nota: La condición de una declaración *if* puede ser cualquier expresión que evalúe *true* (verdadero) o *false* (falso).

Vamos modificando el script del evento *On Click* de nuestro objeto de Botón para que se muestre el mensaje sólo si éste es *"¡Hola Mundo!"*.

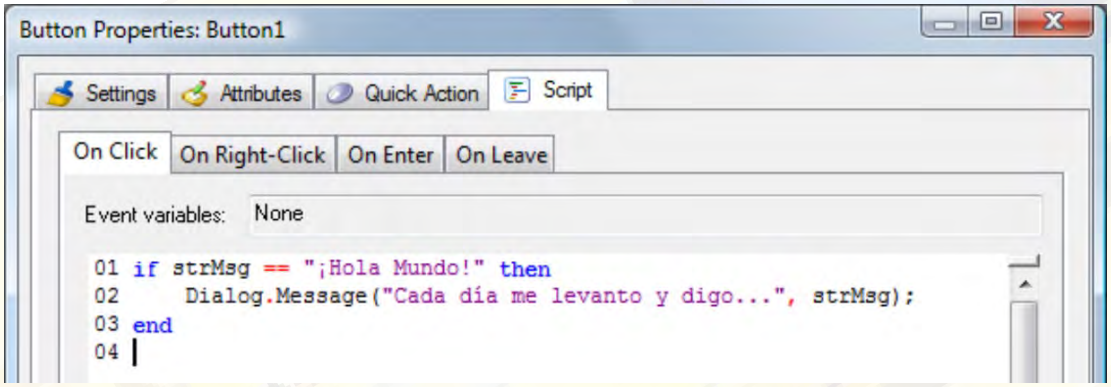
1) Haz doble click en el objeto de Botón. Edita el script *On Click* para añadirle una declaración *if*, tal como la siguiente:

```
if strMsg == "¡Hola Mundo!" then
   Dialog.Message("Cada día me levanto y digo...", strMsg);
end
```

Utiliza un carácter tabulador para sangrar la línea original *"Dialog.Message..."* un poco a la derecha. (Es una buena práctica aplicarle sangría a algunas líneas en el interior de una declaración *if* para ayudar a que sean más fáciles de comprender). Para insertar un carácter tabulador al inicio de la línea, coloca el cursor a la izquierda de la "*D*" y presiona la tecla *Tab* (la tecla que está inmediatamente arriba de la tecla *Bloq mayús*).

Consejo: Puedes sangrar una o más líneas a la vez al seleccionarlas y presionando *Tab* para incrementar el sangrado (a la derecha), o *Shift+Tab* para disminuir el sangrado (a la izquierda).

El script deberá verse como éste cuando lo hayas terminado:



Los dobles signos de igual (==) comparan la igualdad de tal manera que la condición de la declaración *if* sólo será verdadera si *strMsg* contiene "*¡Hola Mundo!*" con exactamente las mismas mayúsculas y ortografía.

Consejo: Una forma fácil de añadir una declaración *if* en el *editor de Script* es seleccionar la línea de texto que quieras colocar en la *if* "interior", hacer click en el botón *Add Code* y luego elegir "*if statement*" del menú. Una "plantilla" de declaración *if* será añadida en torno a la línea que has seleccionado. Después puedes editar la plantilla para ajustarla a tus necesidades.

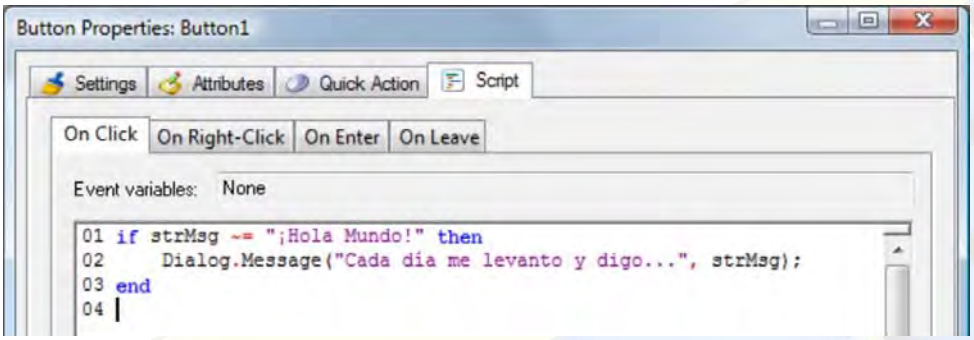
2) Presiona *F5* para previsualizar el proyecto. Cuando se abra la vista previa, haz click sobre el botón para activar el script.

Esta vez no sucede nada porque *strMsg* aún está ajustado a "*¡Buenos Días!*" en el evento *On Preload*. "*¡Buenos Días!*" no es igual a "*¡Hola Mundo!*" así que la condición *if* no se cumple, y el bloque de código entre las palabras clave "*then*" y "*end*" es ignorado completamente.

3) Sal de la vista previa. Edita el script *On Click* del objeto de Botón cambiando el == por ~=, como se muestra a continuación:

```
if strMsg ~= "¡Hola Mundo!" then
   Dialog.Message("Cada día me levanto y digo...", strMsg);
end
```

El script deberá verse como éste cuando lo hayas terminado:



Los signos de tilde e igualdad juntos (~=) comparan la desigualdad de tal manera que la condición de la declaración *if* será verdadera sólo si *strMsg* contiene cualquier cosa distinta de *"¡Hola Mundo!"* comparando exactamente las mismas mayúsculas y ortografía.

Es decir, si por ejemplo a *strMsg* le asignaras el valor de *"¡Hola mundo!"* la desigualdad sí se cumpliría porque *"¡Hola Mundo!"* difiere de *"¡Hola mundo!"* porque *M* no es exactamente igual que *m*.

4) Previsualiza el proyecto. Cuando se abra la vista previa, haz click en el botón.

Esta vez, debido a que *strMsg* contiene *"¡Buenos Días!"*, lo cual definitivamente es desigual a *"¡Hola Mundo!"*, el mensaje sí aparecerá.

Los operadores == y ~= son bien precisos cuando quieres verificar cadenas con una comparación exacta ¿Pero qué tal si no estás seguro de las mayúsculas? ¿Qué tal si el usuario introduce datos y tú no te preocupas de que estén escritos correctamente?

Una solución es utilizar un viejo truco de programación: sólo haz que se conviertan en minúsculas (o en mayúsculas) todas las letras de la cadena desconocida, y haz lo mismo con la cadena que quieres compararla.

Vamos modificando nuestro script para preguntarle algo al usuario en el mensaje, y luego mostrar lo que él escribió, pero sólo si escribe las palabras *"hola mundo"* entre signos de admiración.

5) Sal de la vista previa. Edita el script *On Click* del objeto de Botón de modo que se vea como éste:

```
strMsg = Dialog.Input("", "Introduce tu mensaje:");
if String.Upper(strMsg) == "¡HOLA MUNDO!" then
    Dialog.Message("Cada día me levanto y digo...", strMsg);
else
    Dialog.Message("Mm...", "No escribiste ¡Hola Mundo!");
end
```

La primera línea utiliza una acción *Dialog.Input* para desplegar un mensaje de diálogo con un campo de entrada en el cual el usuario puede escribir. Cualquier cosa que el usuario escriba es entonces asignada a la variable *strMsg*.

Nota: Este nuevo valor reemplaza al valor que le había sido asignado a *strMsg* en el evento *On Preload* de la página.

En la condición de la declaración *if* utilizamos una acción *String.Upper* para convertir el contenido de *strMsg* a caracteres todos en mayúsculas, y luego se compara con "*¡HOLA MUNDO!"*.

Hemos añadido una palabra clave "*else*" después de la primera acción *Dialog.Message*. Esto básicamente divide la declaración *if* en dos partes; la parte "*then*", que sólo sucede si la condición es verdadera; y la parte "*else*", que sólo sucede si la condición es falsa. En este caso, la parte *else* utiliza una acción *Dialog.Message* para indicarle al usuario que no escribió el mensaje correcto.

6) Previsualiza el proyecto. Prueba haciendo click en el botón y escribiendo diferentes mensajes dentro del campo de entrada.

Dependiendo de lo que escribas, podrás ver ya sea el mensaje "*¡hola mundo!*", o bien el mensaje "*No escribiste ¡Hola Mundo!*".

Date cuenta que si escribes alguna variación de *"¡hola mundo!"*, tal como "¡hola mundo!", o "¡hOlA mUnDo!", el uso de mayúsculas que escribas será preservado en el mensaje que aparezca. Aún cuando utilizamos una acción *String.Upper* para convertir la cadena del mensaje toda en letras mayúsculas en la condición de la declaración *if*, el contenido actual de la variable permanece sin cambios.

Cuando la acción *String.Upper* convierte un valor a letras todas en mayúsculas, ésta no cambia el valor de la variable… sólo la retoma, la convierte y la pasa de mano.

7) Sal de la vista previa.

Ahora sabes cómo comparar dos cadenas sin ser melindroso con el uso de mayúsculas. Los programadores llaman a esto una comparación *case-insensitive (insensible a la diferencia entre mayúsculas y minúsculas)*, mientras que una comparación que sólo concuerda con un uso de mayúsculas perfecto es considerada *case-sensitive (sensible a la diferencia entre mayúsculas y minúsculas).*

Consejo: También puedes utilizar una acción *String.CompareNoCase* para ejecutar una comparación *case-insensitive*.

Probando un Valor Numérico

Otro uso común de la declaración *if* es probar un valor numérico para ver si éste ha alcanzado cierta cantidad. Para demostrar esto, vamos creando otro botón que muestre un mensaje después de que hagas click sobre él 5 veces.

1) Añádele otro objeto de Botón a la página.

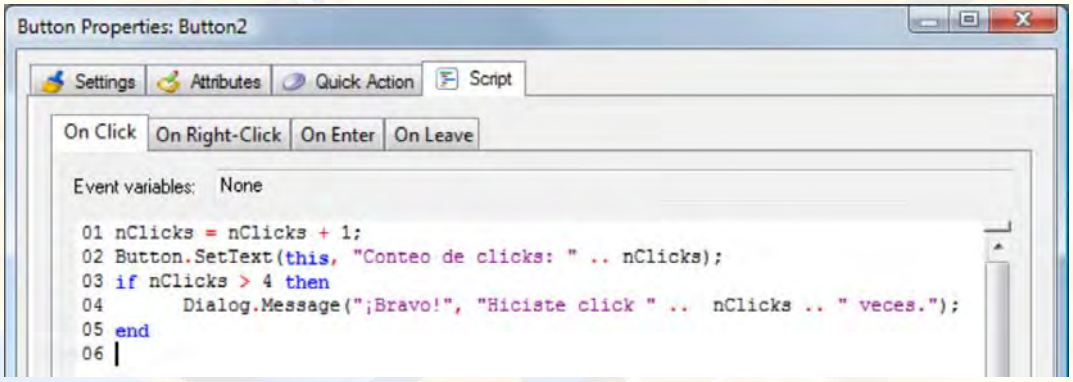
No importa cuál botón elijas, o en qué parte de la página coloques el botón. Sólo necesitamos algo sobre lo cual clickear, de modo que podamos activar un evento *On Click* y ejecutar alguna acción.

2) Agrega el siguiente script en el evento *On Click* del objeto de Botón:

```
nClicks = nClicks + 1;
Button.SetText(this, "Conteo de clicks: " .. nClicks);
if nClicks > 4 then
    Dialog.Message("¡Bravo!", "Hiciste click " ..  nClicks .. " veces.");
end
```

Haz doble click en el nuevo objeto de Botón que añadiste, luego haz click en la pestaña *Script* y escribe el script de arriba dentro del evento *On Click.*

Debería verse así cuando esté terminado:



La primera línea le agrega 1 al valor actual contenido en la variable llamada *nClikcs*. (En el próximo paso le añadiremos algún otro script al evento *On Preload* de la página para ajustar esta variable a 0 cuando se cargue la página).

La segunda línea utiliza una acción *Button.SetText* para cambiar el texto del objeto de Botón a "*Conteo de clicks: n*", donde *n* es el valor actual de *nClicks*.

La acción *Button.SetText* tiene dos parámetros. El primer parámetro es una cadena conteniendo el nombre del objeto sobre el cual quieres operar. En este caso, utilizamos la variable especial *this*, la cual contiene el nombre del objeto sobre el que es activado el evento *On Click*. Básicamente esto es lo mismo que utilizar el nombre del objeto, tal como lo siguiente:

```
Button.SetText("Button2", "Conteo de clicks: " .. nClicks);
```

El segundo parámetro de la acción *Button.SetText* es una cadena conteniendo el texto que quieres mostrar en el objeto de Botón. Utilizamos el *operador de eslabonamiento* para unir el valor actual de *nClicks* al final de la cadena "*Conteo de clicks:* ". Observa

que esta cadena tiene un espacio en blanco al final de ella, así que habrá un espacio entre los dos puntos (:) y el valor de *nClicks*. (De este modo, la cadena resultante se verá mejor).

El *operador de eslabonamiento* consiste en dos puntos seguidos (..). De hecho, con frecuencia es referido como el operador "*punto-punto*". Este es utilizado para unir dos cadenas juntas para formar una nueva cadena combinada. Es algo así como un "*pegamento*" de cadenas.

Nota: Técnicamente, el valor interno *nClicks* no es una cadena (es un número). Sin embargo, realmente esto no importa. Cuando se hace un eslabonamiento, AutoPlay automáticamente convierte el número en la cadena equivalente como se requiera.

El resto del script es sólo una declaración *if* que prueba si el valor *nClicks* es más grande que 4, y muestra un mensaje cuando esto es verdadero. (Un par más de operadores de eslabonamiento son utilizados en la acción *Dialog.Message* para crear la cadena de su segundo parámetro. Observa que también puedes incluir fácilmente el contenido de una variable "*a la mitad*" de una cadena).

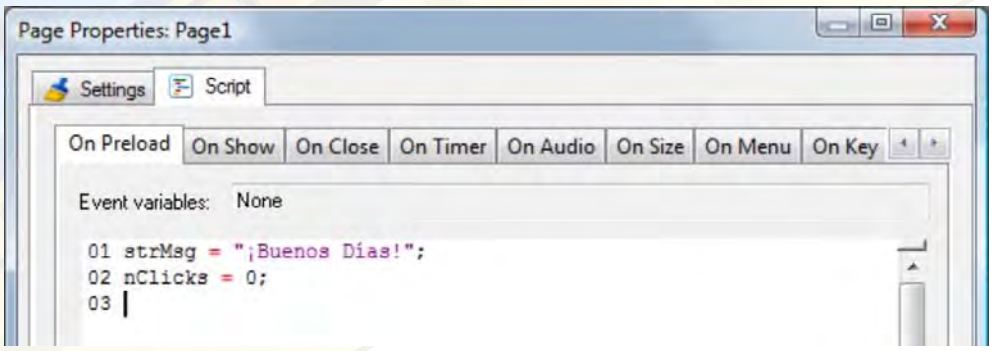
Una vez que hayas terminado la edición del script *On Click* del botón, haz click en *Aceptar* para cerrar el *editor de Script*.

3) Agrega la siguiente línea en el evento *On Preload* de la página:

nClicks = 0;

Necesitamos inicializar *nClicks* antes de que el botón sea clickeado, así que nuestro script en el evento *On Click* del botón iniciará contando desde 0. Date cuenta que no podemos simplemente ajustar *nClicks* a 0 en el evento *On Click* del botón porque sería reajustado a cero cada vez que el botón fuera clickeado.

Para añadirle una línea al evento *On Preload* de la página, sólo haz doble click en la superficie de la página, luego haz click en la pestaña *On Preload* del *editor de Script* y agrégale la línea *nClicks = 0;* al script existente. Una vez que lo hayas hecho, deberá verse así:



No importa si colocas la línea *nClicks = 0*; antes o después de la línea existente… el orden de las líneas individuales en este script no es importante.

4) Previsualiza el proyecto y haz click 5 veces en el objeto de Botón.

Cada vez que hagas click en el botón, su texto cambiará para mostrar cuántas veces has hecho click sobre él. Después de que hayas clickeado 5 veces sobre el botón, aparecerá un mensaje diciéndote que hiciste click 5 veces.

5) Sal de la vista previa.

Muy bien. Acabas de crear un pequeño programa muy sofisticado.

Consejo: Si quieres aventurarte, intenta si puedes modificar el script para detener el anuncio del número de clicks una vez que se hayan alcanzado los 10 clickeos.

Te damos una pista: necesitas utilizar un operador lógico como "*or*" y "*and*" para probar más de una cosa al mismo tiempo en la condición de la declaración *if*. Por ejemplo, podrías hacer que la prueba sólo sea verdadera entre 3 y 7 utilizando una condición como ésta:

```
if (nCount > 2) and (nCount < 8) then
   -- sólo es verdadera si es mayor que 2 y menor que 8
end
```

La segunda línea en el script de arriba es un *comentario* (el cual realmente no hace nada). Todo lo que se escriba en una línea después de dos guiones (--) será completamente ignorado por AutoPlay. Puedes utilizar tales comentarios cuando quieras añadir notas o explicaciones "internas" en tus scripts.

Uso de un Ciclo *For*

Algunas veces es útil poder repetir varias veces un montón de acciones sin tener que escribirlas una y otra vez. Una manera de lograr esto es utilizando un ciclo *for*.

La sintaxis básica del ciclo *for* es:

```
for variable = start, end, step do
      hacer algo aquí
end
```

Por ejemplo:

```
for n = 10, 100, 10 do
   Dialog.Message("", n);
end
```

Este script simplemente cuenta desde 10 hasta 100, de 10 en 10, mostrando el valor actual de la variable *n* en un mensaje de cuadro de diálogo. Esto logra el mismo propósito que si escribes:

```
Dialog.Message("", 10);
Dialog.Message("", 20);
Dialog.Message("", 30);
Dialog.Message("", 40);
Dialog.Message("", 50);
Dialog.Message("", 60);
Dialog.Message("", 70);
Dialog.Message("", 80);
Dialog.Message("", 90);
Dialog.Message("", 100);
```

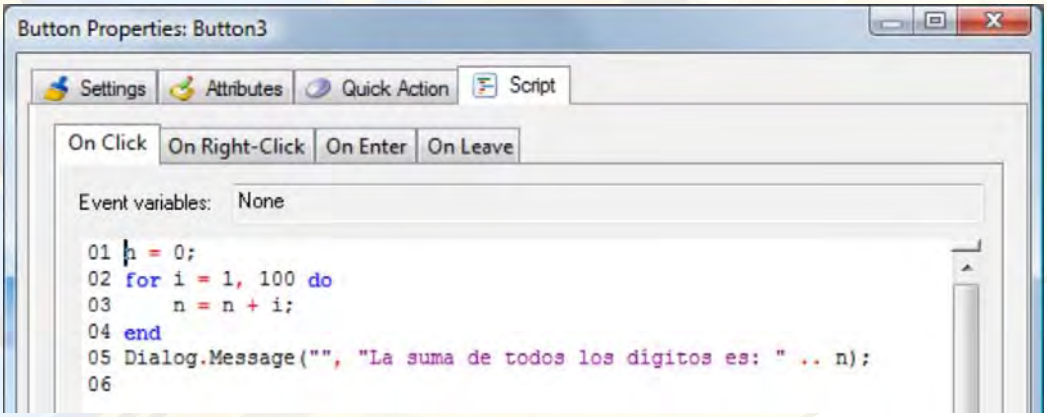
Obviamente, el ciclo *for* hace mucho más fáciles las acciones de repetición similar.

Vamos utilizando un ciclo *for* sencillo para sumar todos los dígitos entre 1 y 100, y mostrar el resultado.

1) Agrega un objeto de Botón en la página. Añáde el siguiente script al evento *On Click* de este botón.

```
n = 0;
for i = 1, 100 do
    n = n + i;
end
Dialog.Message("", "La suma de todos los dígitos es: " .. n);
```

El script debería verse así cuando esté terminado:



La primera línea crea una variable llamada *n* y la ajusta a *0*. La siguiente línea le ordena al ciclo *for* contar de 1 a 100, almacenando la "cuenta" actual en cada etapa en una variable llamada *i*.

Durante cada "paso" a través del ciclo, el script entre el "*do*" y el "*end*" será ejecutado. En este caso, éste consiste en una única línea que suma los valores actuales de *n* y de *i*, y luego almacena el resultado de nuevo en *n*. En otras palabras, le agrega *i* al valor actual de *n*.

Este ciclo *for* es lo mismo que escribir:

n = n + 1;
n = n + 2;
n = n + 3;
n = n + 4;

...hasta llegar a 100.

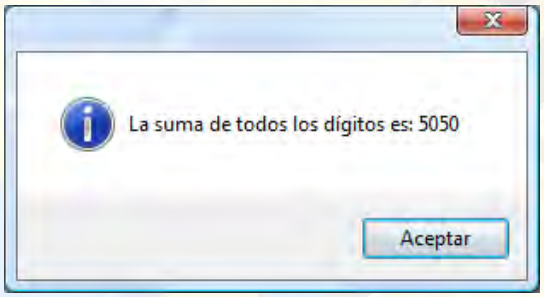
La última línea de nuestro script simplemente le muestra al usuario el resultado del cálculo en un mensaje de cuadro de diálogo.

Consejo: Puedes utilizar el botón *Add Code* para insertar un ejemplo de un ciclo *for*, completando la sintaxis con comentarios explicativos. Luego puedes editar el ejemplo para ajustarlo a tus propias necesidades.

2) Previsualiza en proyecto y haz click en el objeto de Botón.

Cuando hagas click en el objeto de Botón, el ciclo *for* ejecutará a la velocidad de la luz el cálculo 100 veces, y luego la acción *Dialog.Message* mostrará el resultado.

Todo esto sucede muy rápido.



3) Sal de la vista previa.

Probablemente no utilizarás los ciclos *for* con la frecuencia que utilizarás las declaraciones *if*, pero definitivamente vale la pena saber cómo se utilizan. Cuando necesites repetir pasos, éstos pueden ahorrarte mucho esfuerzo.

Por supuesto que cuando hablamos de ahorrar esfuerzo, las verdaderas campeonas son las funciones.

Creación de Funciones

Una *función* es sólo una porción de script que puedes definir, nombrar y luego llamar desde cualquier otro lado.

Puedes utilizar *funciones* para evitar la repetición del mismo script en diferentes lugares, básicamente creando tus propias "acciones" personalizadas (pudiendo completarlas con parámetros y valores de retorno si así lo quieres).

En general, las funciones están definidas como:

function *function_name* (*arguments*)
    function script here
    return *return_value*;
end

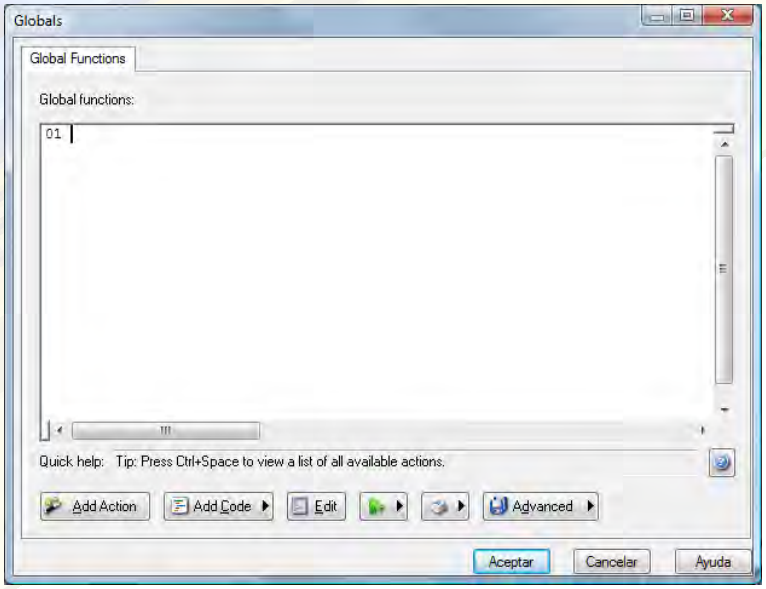
La palabra clave "*function*" le indica a AutoPlay que lo que sigue es una definición de función. La *function_name* es simplemente un nombre único para la función. La parte *arguments* es la lista de parámetros que pueden ser pasados a la función cuando ésta sea llamada. Esta es básicamente una lista de nombres de variable que recibirán los valores que sean pasados. (Las variables resultantes son locales a la función, y sólo tienen significado dentro de ella). Una función puede tener tantos argumentos como quieras (incluso ninguno).

La palabra clave "*return*" le indica a la función que devuelva uno o más valores al script que la llamó.

La forma más fácil de aprender acerca de las funciones es probar algunos ejemplos, así que sumerjámonos en ello.

1) Elige *Project > Global Functions.*

Esto abre el cuadro de diálogo *Global Functions*



El cuadro de diálogo *Global Functions* es el lugar conveniente para colocar cualquier función o variable que quieras tener disponible a lo largo de todo tu proyecto. Cualquier script que añadas en este cuadro de diálogo será ejecutado cuando tu aplicación sea lanzada, justo antes de que el evento *On Startup* sea activado.

2) Añade el siguiente script:

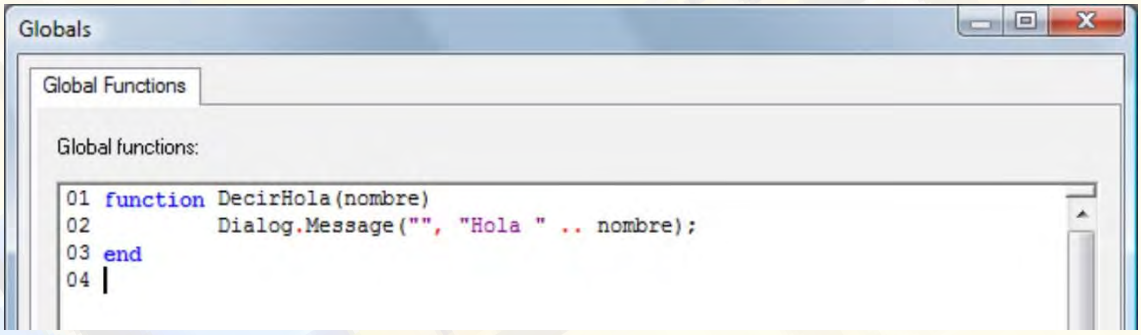
```
function DecirHola(nombre)
    Dialog.Message("", "Hola " .. nombre);
end
```

Cuando hayas terminado, haz click en *OK* para cerrar el cuadro de diálogo.

Este script define una función llamada *DecirHola* que toma un solo argumento (el cual hemos llamado "*nombre*") y muestra un mensaje simple.

Date cuenta que esto solamente define la función. Cuando esté script sea ejecutado, "cargará" la función dentro de la memoria, pero realmente no mostrará el mensaje sino hasta que la función sea llamada.

Deberá verse así cuando lo hayas terminado:



Una vez que has introducido la definición de la función, haz click en *Aceptar* para cerrar el cuadro de diálogo *Global Functions*.

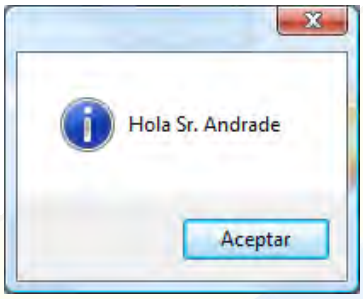
3) Añade un objeto de Botón a la página y agrégale este script en su evento On Click:

```
DecirHola("Sr. Andrade");
```

Este script llama a la función *DecirHola* que fue definida en el cuadro de diálogo *Global Functions*, pasando la cadena "*Sr. Andrade*" como el valor para el parámetro "*nombre*" de la función.

4) Previsualiza el proyecto y haz click en el objeto de Botón.

Cuando haces click en el objeto de Botón, el script del evento *On Click* llama a la función *DecirHola*, la cual entonces muestra su mensaje.



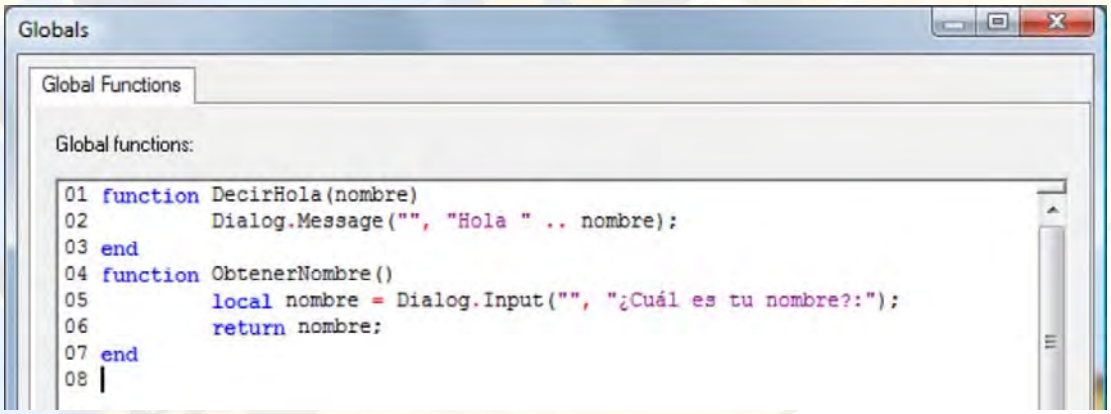
Date cuenta que la función *DecirNombre* fue capaz de utilizar la cadena que le pasamos y la mostró en el mensaje.

5) Sal de la vista previa. Elige *Project* > *Global Functions* y añade el siguiente script debajo de la función *DecirNombre*:

```
function ObtenerNombre()
    local nombre = Dialog.Input("", "¿Cuál es tu nombre?:");
    return nombre;
end
```

Cuando lo hayas hecho, haz click en *Aceptar* para cerrar el cuadro de diálogo.

El resultado final deberá verse como éste:



Este script define una función llamada *ObtenerNombre* que no toma ningún parámetro. La primera línea en el interior de la función utiliza una acción *Dialog.Input* para mostrar un diálogo de mensaje con un campo de entrada en él, pidiéndole al usuario que introduzca su nombre. El valor devuelto de esta acción (es decir, el texto que el usuario escriba) es entonces almacenado en la variable local llamada *nombre*.

La palabra clave "*local*" hace que la variable sólo exista en el interior de esta función. Es básicamente como decir: "para el resto de esta función, en cualquier momento que

utilice "*nombre*" me estoy refiriendo a una variable local temporal, incluso si hay una variable global "*nombre*", la cual es posible que exista". El uso de variables locales dentro de funciones es una buena idea (esto previene que cambies el valor de una variable global sin tener la intención).

La segunda línea en el interior de la función devuelve el valor actual de la variable local "*name*" al script que llamó la función.

Consejo: De hecho podríamos hacer que este script de la función se ajuste a una sola línea, librándose completamente de la variable. En lugar de almacenar el valor devuelto de la acción *Dialog.Input* en una variable, y luego devolver el contenido de esa variable, simplemente podríamos poner juntas esas dos declaraciones de la siguiente manera:

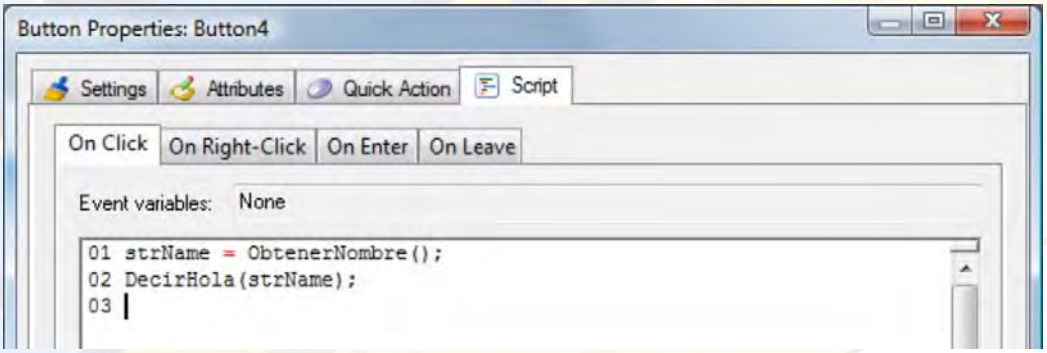
```
function ObtenerNombre()
   return Dialog.Input("", "¿Cuál es tu nombre?:");
end
```

Esto haría que la función *ObtenerNombre* retornara el valor que fue devuelto de la acción *Dialog.Input*, sin almacenarlo primero en una variable.

6) Edita el script en el evento *On Click* del objeto de Botón de modo que, en su lugar, se vea como:

```
strName = ObtenerNombre();
DecirHola(strName);
```

Deberá verse así cuando hayas terminado:

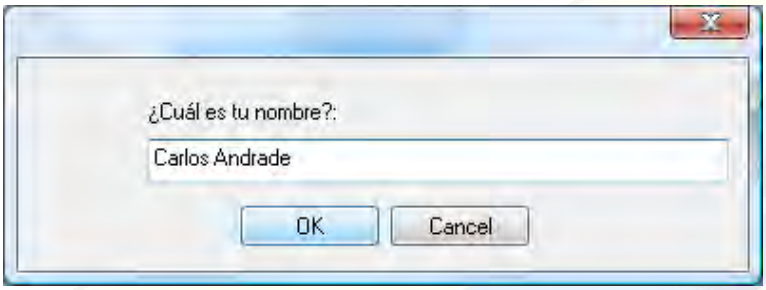


La primera línea llama a nuestra función *ObtenerNombre* para solicitarle al usuario su nombre, y luego almacena el valor devuelto de la variable *ObtenerNombre* en una variable llamada *strName*.

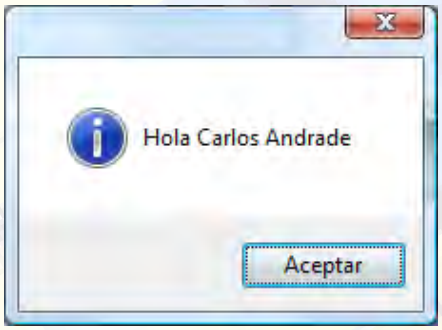
La siguiente línea pasa el valor en *strName* a nuestra función *DecirHola*.

7) Previsualiza el proyecto. Prueba el script haciendo click en el objeto de Botón.

Cuando hagas click en el objeto de Botón, un cuadro de diálogo de entrada aparecerá, pidiéndote introducir tu nombre.



Después de que escribas tu nombre y hagas click en *OK* (o presiones *Enter*), un segundo cuadro de diálogo aparecerá, saludándote con el nombre que introdujiste.



¿Fantástico verdad?

8) Sal de la vista previa. Edita el script en ese mismo evento *On Click* del objeto de Botón de tal manera que se vea como:

DecirHola(ObtenerNombre());

Esta versión de script elimina por completo la variable *strName*. En lugar de eso, utiliza el valor devuelto de la función *ObtenerNombre* como el argumento para la función *DecirHola*.

En otras palabras, pasa el valor devuelto de la función *ObtenerNombre* directamente a la función *DecirHola*.

Cuando una función devuelve un valor, puedes utilizar una llamada de la función de la misma forma que si utilizaras el valor, o una variable que contiene el valor. Esto te permite utilizar el valor devuelto de una función sin tener que proponer un nombre único para una variable temporal.

9) Previsualiza el proyecto y de nuevo prueba el script. Cuando hayas terminado, sal de la vista previa.

El script deberá trabajar exactamente igual que antes: se te preguntará tu nombre y luego serás saludado con él.

Este es sólo un ejemplo sencillo, pero te dará una idea de lo increíblemente poderosas que son las características de las funciones. Con ellas puedes condensar grandes piezas de script dentro de simples llamadas de función, las cuales son mucho más fáciles de escribir y te dan una sencilla localización central para hacerle cambios a ese script. También te permite crear flexibles "*subrutinas*" que aceptan diferentes parámetros y devuelven resultados, igual que las acciones incorporadas en el AutoPlay.

Y a pesar de todo ese poder, son realmente muy sencillas de utilizar.

¿A Dónde Ir a Partir de Aquí?

Bueno, ésta es la última Lección. Esperamos que hayas disfrutado este entrenamiento en AutoPlay y que esta Guía del Usuario haya sido útil y práctica.

El siguiente capítulo contiene una guía más detallada del lenguaje de programación del AutoPlay.

Tómate la libertad de conectarte con usuarios de AutoPlay Media Studio en nuestros foros en línea, donde abunda la asistencia y camaradería. Puedes hacerlo eligiendo *Help > User Forums* directamente desde el menú del programa AutoPlay.

Mientras estés ahí, asegúrate de revisar la referencia del programa en línea y también la útil lista de preguntas "*¿How do I...?*". (También puedes acceder a estos recursos directamente desde el menú *Help* del AutoPlay).

Hay mucho más que puedes aprender acerca de este producto si alguna vez lo necesitas o lo deseas. Las oportunidades ahí están, si hay algo que quieras hacer con el AutoPlay, hay por lo menos una forma de conseguirlo, y probablemente más.

Como dijo una vez uno de nuestros usuarios más instruidos: *AutoPlay es fácil de aprender, pero difícil de dominar.*

Resumen de la Lección 10

En esta lección aprendiste cómo:

- Mostrar un mensaje
- Utilizar una variable
- Añadir una declaración *if*
- Probar valores numéricos
- Ajustar el texto de un objeto de Botón
- Eslabonar (unir) cadenas
- Comparar cadenas

- Utilizar un ciclo *for*
- Crear funciones