



Script Component

Using :

A script is a set of instructions or commands written in a specific programming language "Pascal" or "C++" or "Basic" or "Java" syntax.

Events :

?

Commands :

Load	Load Script From File
Run	Start Script
Stop	Stop Script

Functions :

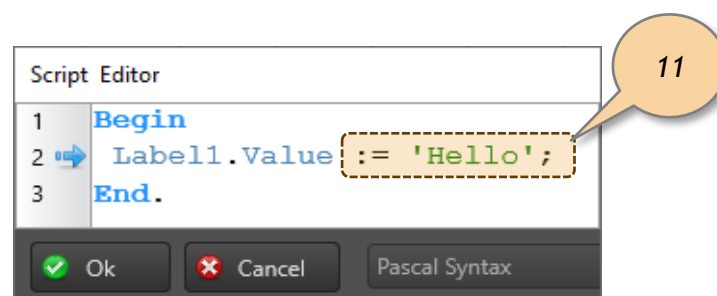
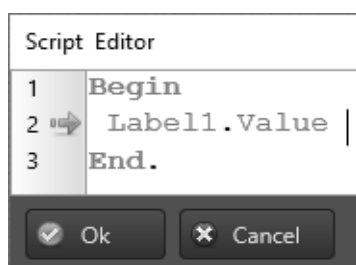
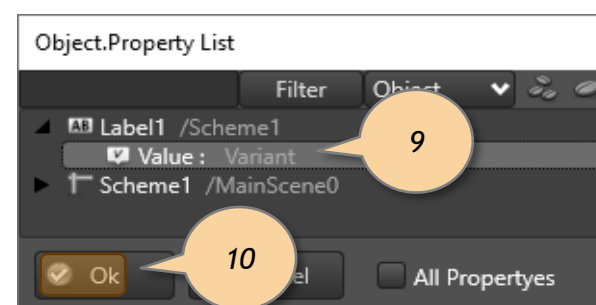
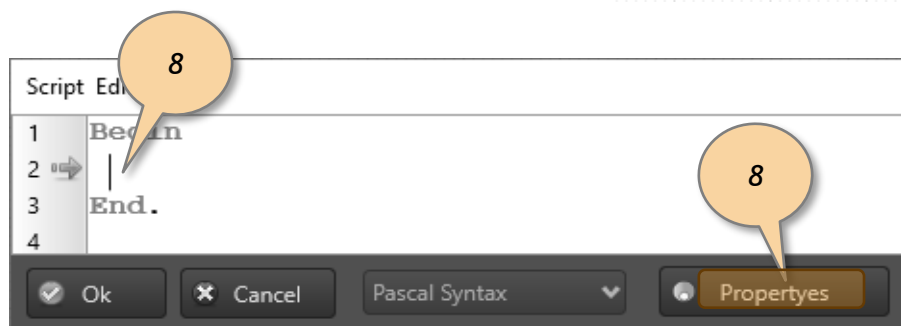
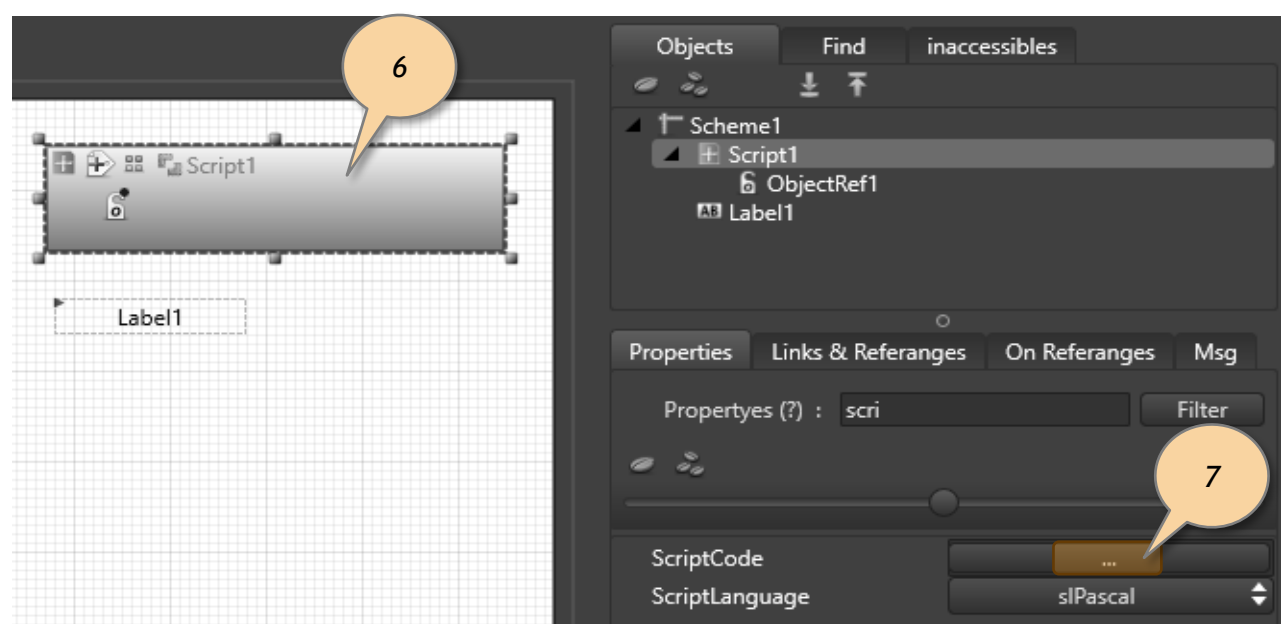
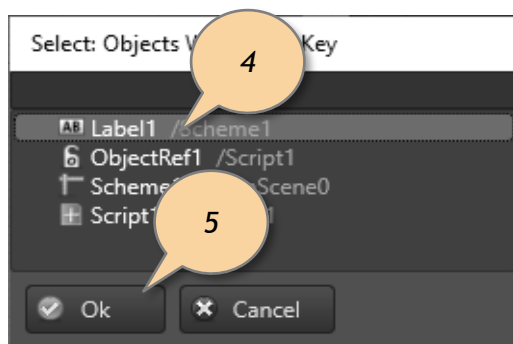
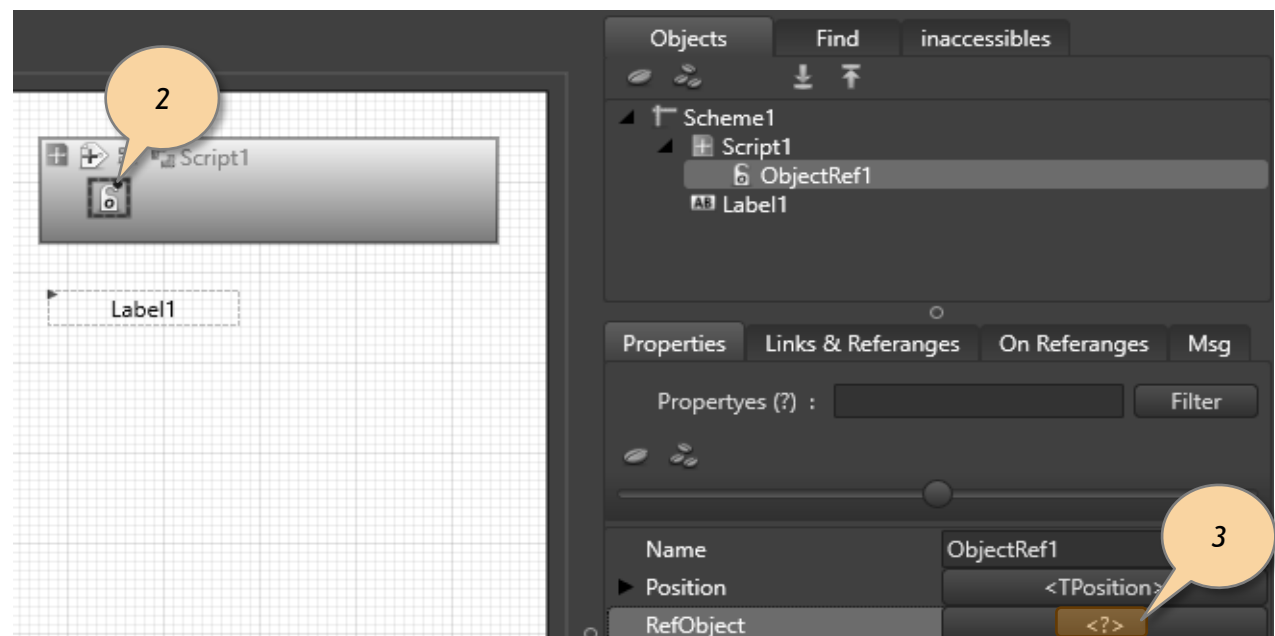
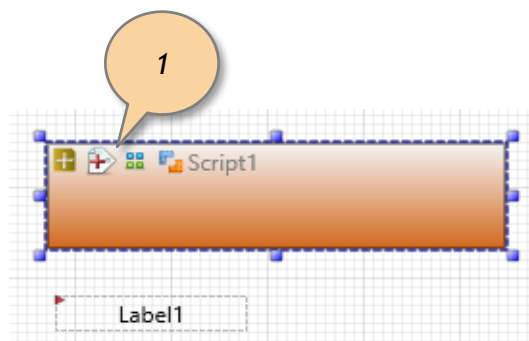
?

Propertyes :

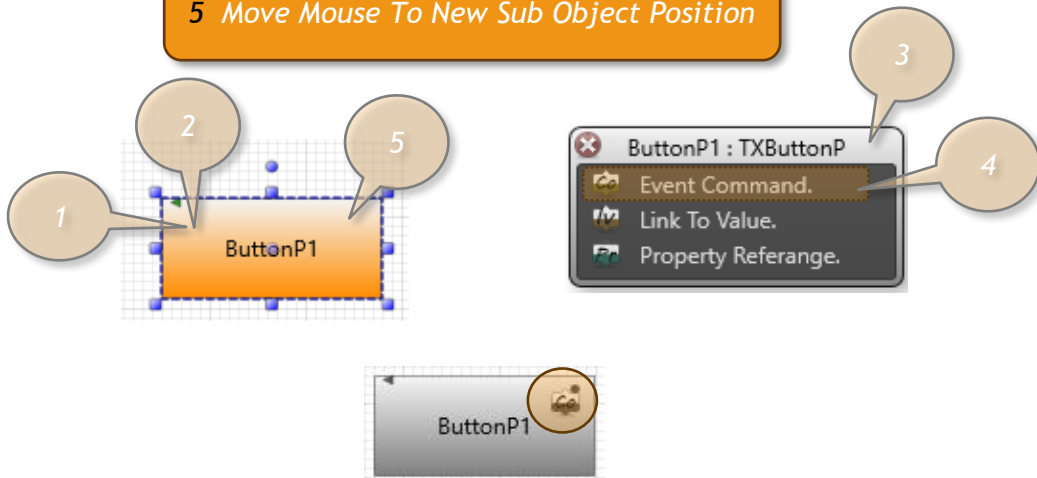
FileName	Script File Name
FilePath	Script File Path
ScriptCode	Script Code Text
ScriptLanguage	Script Programming Language
AutoRefObjectRename	When the user changes the object name, it also changes in the script.

Note :

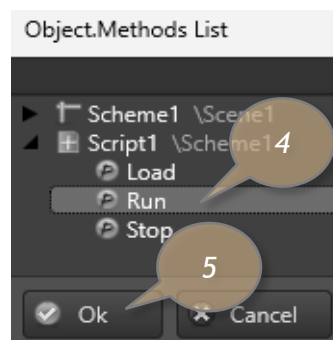
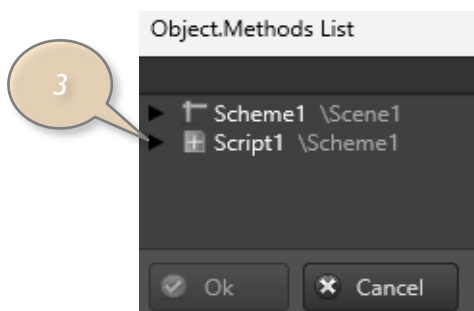
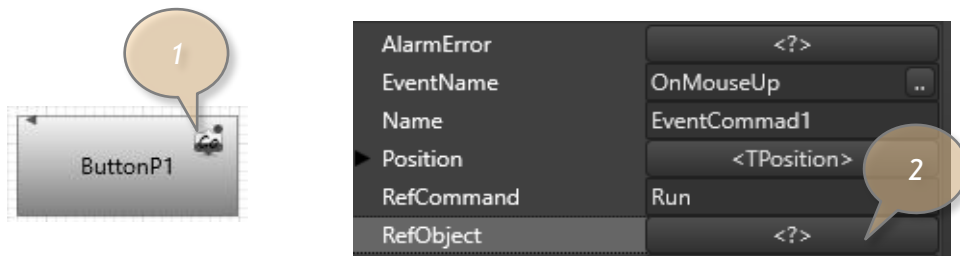
Example



- 1 Select Object
- 2 Click Right Mouse Button
- 3 Show PopUp
- 4 Select Item
- 5 Move Mouse To New Sub Object Position



- 1 Select Object
- 2 Click *Button1.RefObject* Property Button
- 3 Expand *Script1* Object Methods
- 4 Select *Script1.Run* Method



Structure

Java structure:

```
// the "inport" should be located before any other chapter
inport 'unit1.js', 'unit2.js'

var i, j = 0;

# define pi = 3.14159

function p1()    // procedures and functions
{               // no nested procedure
}

p1();    // main procedure.
for ( i = 0 ; i < 10 ; i++ ) j++;
```

C++ structure:

```
// the "include" should be located before any other chapter
# include 'unit1.cpp', 'unit2.cpp'

int i, j = 0;

# DEFINE pi = 3.14159

void p1()    // procedures and functions
{           // no nested procedure
}

{ // main procedure.
  p1(); // call p1
}
```

PascalScript structure:

```
program MyProgram;    // optional

// the "uses" should be located before any other chapter
uses 'unit1.pas', 'unit2.pas';

var
i, j: Integer;

const
pi = 3.14159;

procedure p1;    // procedures and functions
var
i: Integer;

procedure p2; // nested procedure
begin
end;

begin // p1
  p2; // call p2
end;

begin // main procedure.

  For i := 0 To 10 Do
  begin
    j := j + 1;
  end;
  p1; // call p1

end.
```

Basic structure:

```
// the "inport" should be located before any other chapter
inport 'unit1.vb', 'unit2.vb'

dim i, j = 0

Function p1()    // procedures and functions
{               // no nested procedure
}

For i = 0 To 10
  j = j + 1
Next

p1() // call p1
```

Object

PascalScript

Using Objects Property in the script

```
Rectangle1.Fill.Color := claRed;
```

C++

Using Objects Property in the script

```
Rectangle1.Fill.Color = claRed;
```

PascalScript

Using String Variable in the script

```
var Variable: String;  
begin  
  Variable := 'Hello!';  
end.
```

C++

Using String Variable in the script

```
string Variable;  
{  
  Variable = "Hello!";  
}
```

Types :

Byte
Word
Integer
Longint
Cardinal
TColor
TAlphaColor
Boolean
Real
Single
Double
Extended
TDate
Ttime
TDateTime
Char
String
Variant
Array

Constants :

True :Boolean
False :Boolean

Functions And Procedures :

function IntToStr(i: Integer): String
function FloatToStr(e: Extended): String
function DateToStr(e: Extended): String
function TimeToStr(e: Extended): String
function DateTimeToStr(e: Extended): String
function BoolToStr(B: Boolean): string
function VarToStr(v: Variant): String
function StrToInt(s: String): Integer
function StrToInt64(s: String): Int64
function StrToFloat(s: String): Extended
function StrToDate(s: String): Extended
function StrToTime(s: String): Extended
function StrToDateTime(s: String): Extended
function StrToBool(const S: string): Boolean
function Format(Fmt: String; Args: array): String

function FormatFloat(Fmt: String; Value: Extended): String
function FormatDateTime(Fmt: String; DateTime: TDateTime): String
function FormatMaskText(EditMask: string; Value: string): string
function EncodeDate(Year, Month, Day: Word): TDateTime
procedure DecodeDate(Date: TDateTime; var Year, Month, Day: Word)
function EncodeTime(Hour, Min, Sec, MSec: Word): TDateTime
procedure DecodeTime(Time: TDateTime; var Hour, Min, Sec, MSec: Word)
function Date: TDateTime
function Time: TDateTime
function Now: TDateTime
function DayOfWeek(aDate: TDateTime): Integer
function IsLeapYear(Year: Word): Boolean
function DaysInMonth(nYear, nMonth: Integer): Integer
function Length(s: Variant): Integer
function Copy(s: String; from, count: Integer): String
function Pos(substr, s: String): Integer
procedure Delete(var s: String; from, count: Integer)
procedure DeleteStr(var s: String; from, count: Integer)
procedure Insert(s: String; var s2: String; pos: Integer)
function Uppercase(s: String): String
function Lowercase(s: String): String
function Trim(s: String): String
function NameCase(s: String): String
function CompareText(s, s1: String): Integer
function Chr(i: Integer): Char
function Ord(ch: Char): Integer
procedure SetLength(var S: Variant; L: Integer)
function Round(e: Extended): Integer
function Trunc(e: Extended): Integer
function Int(e: Extended): Integer
function Frac(X: Extended): Extended
function Sqrt(e: Extended): Extended
function Abs(e: Extended): Extended
function Sin(e: Extended): Extended
function Cos(e: Extended): Extended
function ArcTan(X: Extended): Extended
function Tan(X: Extended): Extended
function Exp(X: Extended): Extended
function Ln(X: Extended): Extended
function Pi: Extended
procedure Inc(var i: Integer; incr: Integer = 1)
procedure Dec(var i: Integer; decr: Integer = 1)
procedure Randomize
function Random: Extended
function ValidInt(clnt: String): Boolean
function ValidFloat(cFlt: String): Boolean
function ValidDate(cDate: String): Boolean
function ExtractFilePath(const FileName: string): string
function VarArrayCreate(Bounds: Array; Typ: Integer): Variant

```
function VarType(V: Variant): Integer
```