

# Introduction to Scripting in E-Prime 2.0

---

- 1) Quick E-Prime concept review
  - a. Scope – Lists and procedures
    - i. When trying to resolve, search can see higher levels, not lower
  - b. Attributes (in Lists)
    - i. Major independent or control variables
  - c. Object Properties
    - i. Property pages
  - d. Generate PreRun
    - i. Set to 'Before Object Run' while learning script
  - e. PreRelease
    - i. Be careful/turn off before an InLine script
  - f. Save your work often!
- 2) What is scripting and E-Basic?
  - a. Scripting is writing commands to an existing program like E-Prime.
    - i. E-Prime itself is made out of code.
  - b. E-Prime understands E-Basic, our programming language
  - c. Designed to be friendly to beginners, based on VBA
  - d. Learning a programming language to be a bit like learning another spoken language
    - i. Learn some conventions and new concepts
    - ii. Learn how phrases are constructed (syntax)
  - e. What is a Graphical User Interface? (GUI)
  - f. Who will be reading your script?
    - i. Colleagues who don't know anything about your experiment
    - ii. You years from now when you've forgotten the particulars
- 3) Help is available
  - a. Web support site, [support.pstnet.com](http://support.pstnet.com)
  - b. E-Basic Help
  - c. User's Guide
  - d. VBA for Dummies
- 4) The scripting process
  - a. Flow chart/design phase
    - i. Think of it in logical pieces rather than sum of its parts
    - ii. Don't try to do too much at once
    - iii. Start small and build on a solid foundation
  - b. Implement first segment
  - c. Test the segment
  - d. Debug the first segment
    - i. Find and remove errors

- ii. Debugging tips will come later
  - e. Repeat until complete: implement, test, debug the next segment
  - f. Keep testing
    - i. Run a few pilot subjects
- 5) Change a property of an object using script
  - See IntroToScripting\_1\_ChangeObjectProperty.es2
  - a. Compare to GUI property page
  - b. Example
    - i. Stimulus currently says "This text will not appear."
    - ii. Add In Line in scope
      - 1. Rename to ChangeText
    - iii. Add comment to explain what you are writing
      - 1. Comments are helpful to you and others reading the script
      - 2. Use comments to outline your ideas before writing script
      - 3. You may write anything you want here
      - 4. Two ways to comment:
        - a. Single line '
        - b. Multiple line /\* \*/
    - iv. Add script
      - 1. Stimulus.Text = "Hello World!"
        - a. '=' is assignment not equals in this situation
    - v. Run experiment to see that text has been replaced
    - vi. Know that you have changed this for the rest of the experiment as if you had changed the GUI setting.
      - 1. This is different from scope!
- 6) Beyond the GUI properties: adding functionality with Variables and Attributes
  - See IntroToScripting\_2\_Attributes.es2
  - a. Attributes
    - i. Subject to scope
    - ii. Attributes are in the Context object
      - 1. Context is a "behind the scenes" object that you don't see in E-Studio's interface
      - 2. It stores the experiment data and associated routines
      - 3. By default, this object is referred to as 'c' in script
    - iii. The Context object is accessed via Methods (Commands or Functions)
      - 1. Methods cause objects to perform certain actions
      - 2. Some require parameters
        - a. Input that the method will use internally
        - b. See E-Basic help for a list of parameters for each method
    - iv. Accessing already-defined attributes from Lists, script
      - 1. c.GetAttrib("AttributeName")
    - v. Adding new attribute to the context or updating existing one

1. Once an attribute is added to the context, it will be in the data file
  2. `c.SetAttrib "AttributeName", "ContentsOfAttribute"`
- vi. Example:
1. Changing text using attribute references instead of property
  2. Show Stimulus, List attributes
  3. This only works with `GeneratePreRun – BeforeObjectRun`
- b. Temporary storage of values within script - Variables  
See `IntroToScripting_3_Variables.es2`
- i. Keep in mind scope
    1. User script
    2. InLine objects in correct procedure/location
  - ii. Choose descriptive names
    1. Easier to read, reduces logic errors
  - iii. Dim... As statement
    1. Declares variable local to that scope
  - iv. Choose a variable type (see Users Guide for more types and details):
    1. Boolean
      - a. True (1) or False (0)
        - i. Default when initialized: False
    2. Integer
      - a. Math may be performed on these numbers
      - b. Whole numbers
        - i. Default when initialized: 0
        - ii. Range: -32767 to 32767
    3. Long
      - a. Math may be performed on these numbers
      - b. Larger whole numbers
        - i. Default when initialized: 0
        - ii. Range: -2,147,483,648 to 2,147,483,647
      - c. Safer to use when tracking experiment times, etc.
        - i. Prevents "overflow" error due to running out of space
    4. String
      - a. Sequences of characters (words, numbers)
        - i. Default when initialized: "" (aka nothing, an 'empty String')
      - b. Even if values are numbers, math may not be performed
    5. See User's Guide for conversion between types
  - v. Better to use abstract variable than "hard coded" values – easier to understand and update later
  - vi. Example:
    1. Create a counter
    2. Requires script in 3 places due to scope

- a. Dim in user script
- b. Initialize at top of session proc
- c. Increment within procedure
- 3. Discuss Variable on both sides of the equation
- 4. Whitespace is important (no semi colons, etc.)
  - a. To continue a line, use underscore
- 5. Save data to the data file using an attribute

## 7) Debugging

- a. Looking at error messages and full script to find problem spots
- b. Try running in fixed order (if List is random)
- c. Debug.Print
  - i. Command (method) of the Debug object
  - ii. Useful for getting the value of a variable while the experiment is running
  - iii. Seen within Debug tab of Output
  - iv. Requires concatenation:
    - 1. Debug.Print "This variable is " & variablename
    - 2. Spaces are not automatically entered, we must type them
  - v. Example also in IntroToScripting\_3\_Variables.es2
    - 1. Debug.Print the trialCounter each time the TrialProc runs
    - 2. Placement is important since variables change at assignment

## 8) Review: differences between similar concepts

- a. All have different functionality
- b. All have different syntax/parameter requirements
- c. Variables vs. Attributes
  - i. A Variable
    - 1. Holds temporary values
    - 2. Is not logged in data file
    - 3. Created in script
    - 4. Accessible only in current scope/procedure
    - 5. Used in script only
    - 6. Is initialized with the 'Dim' statement (e.g., Dim Counter as Integer)
  - ii. An Attribute
    - 1. Holds values of major independent or control variables
    - 2. Logged in data file
    - 3. Created using List or script
    - 4. Accessible only at current or lower levels of context
    - 5. Used in script or by objects
    - 6. Is initialized either in a List or with the SetAttrib statement (e.g., c.SetAttrib "Counter", 1)
- d. Properties, Methods, Commands, Functions
  - i. Properties

1. Store information regarding the behavior or physical appearance of the object.
2. Referenced using the object.property syntax.
- ii. Methods
  1. Allow access to routines, code that is pre-written to perform an action.
  2. You just need to know the input and output of each method, the details of the code in the routine doesn't need to be known.
  3. Methods are commands or functions
    - a. Commands
      - i. Type of method
      - ii. Actions the object can perform
      - iii. Does not return a value
      - iv. May or may not accept parameters
      - v. Not all commands are available to all objects
        1. See E-Basic Help
      - vi. Referenced using the object.command syntax.
    - b. Functions
      - i. Type of method
      - ii. May or may not accept parameters
      - iii. Returns a value
      - iv. Referenced using the object.function syntax

## 9) Flow Control: If...Then, Loops, and the GoTo statement

See IntroToScripting\_4\_ConditionalStatements.es2

- a. Conditional statements
  - i. If...Then
  - ii. Elseif...Then
  - iii. End If
  - iv. Also Select Case – see User's Guide
  - v. Logical operators
    1. > Greater than
    2. < Less than
    3. = Equal to
      - a. Different from assignment!
    4. >= Greater than or equal to
    5. <= Less than or equal to
    6. <> Not equal to
    7. And
      - a. Returns True If both expressions are true
    8. Or
      - a. Returns True If either expression is true
    9. Xor
      - a. Returns True If only one expression is true.



1. Need to update each one to avoid errors
  - iv. Compile/Run to catch mistakes
  - v. Run a pilot subject to be sure everything works
- c. Try reversing the logic of IntroToScripting\_5\_LoopsAndGoto.es2
  - i. Red car skips picture
  - ii. Blue car waits 3 seconds then shows picture (BlueCar.bmp provided)
- d. Have us help you in support
  - i. Always submit something! Once we see what you are attempting in script, we can help you make it work.

## Q&A