The Workspace

All kinds of commands related to the workspace.

ABOUT

A user-defined procedure to display information about your program.

Syntax

```
ABOUT
```

Description

ABOUT is a user-defined procedure that Logo calls when the user selects the “About...” menu item. To display information about your program, write this procedure and let it either PRINT some informative text, or display an ALERT box containing this information.

Example

```
TO ABOUT
    IGNORE ALERT "|This is my precious app!|
END
```

ALIAS

Defines alias names.

Syntax

```
ALIAS old-name new-name
```

Description

ALIAS defines a new procedure name with the same meaning as an existing procedure name. The new name is an alias name for the existing name and behaves exactly like that name. Values or properties stored at a name are not aliased.

ALIAS is handy for defining shortcuts or translations of commands.

BYE (QUIT,EXIT)

Ends Logo.

Syntax

```
BYE
```

Description

BYE ends the Logo session and logs you out. The command names QUIT and EXIT perform the same function.

CHAR

Converts a number into a Unicode character.

Syntax

```
CHAR number
```

Description

CHAR reports the character whose Unicode code is the input. The input number can be from 0 through #H10FFF. To output the Unicode code corresponding to an input character, use UNICODE.

Use caution when saving a Logo name or procedure that contains unprintable characters, like characters with a value of less than 32. Logo saves them as-is without escaping them. If you, for example, save a value that contains a newline character (CHAR 10), then Logo may not be able to read that file correctly.

Example

```
CHAR 20013
Result: 中
```

CLEARTEXT (CT)

Clears the Listener pane.

Syntax

```
CLEARTEXT
```

Description

CLEARTEXT clears all text and places the cursor in the upper left corner of the Listener pane. CLEARTEXT is the same as the command TYPE CHAR 12.

Example

```
REPEAT 5 [PR [GOOD EVENING]]
CLEARTEXT
```

DATE

Reports the date.

Syntax

```
DATE
```

Description

DATE reports the current date as a four-element list in the form [day month year dayofweek]. The dayofweek is a number from 1 to 7, starting at Sunday.

Example

```
DATE
Result: [21 11 2019 5]
```

FULLSCREEN (FS)

Switches to the Full Screen perspective

Syntax

```
FULLSCREEN
```

Description

FULLSCREEN minimizes the Output panel and maximizes the Graphics canvas.

Example

```
FS
```

HELP

Displays help for a command.

Syntax

```
HELP commandname
```

Description

HELP displays the help file for the requested command.

Example

```
HELP "FORWARD
```

LMAKE (LOCALMAKE)

Create and set a local name.

Syntax

```
LMAKE varname value
```

Description

LMAKE works in two ways. The first time `LMAKE varname value` is used in a procedure, Logo creates a new local name `varname` and assigns `value` to the local `varname`. This is the same as `LOCAL varname MAKE varname value`. For all subsequent uses of `LMAKE varname value`, Logo just uses the existing local `varname`. This the same as `MAKE varname value`. The important thing to consider about LMAKE is that it ensures that the variable being referenced is local (by either creating a new local variable or using an existing local variable).

See also `LOCAL`, `MAKE` and `THING`.

LOAD

Loads a file into Logo.

Syntax

```
LOAD filename
LOAD filename.ext
LOAD [extensions-list]
(LOAD)
```

Description

LOAD transfers the contents of the file specified by its input from the Workspace area to the Logo workspace. The entire file is treated as though it were typed from the keyboard. LOAD throws an error if the file is not successfully loaded.

If no file name extension is specified, LOAD loads the file filename.LGO. To load a file that has no extension, a period is necessary after the filename.

If you use a list of file extensions as the file name, or use the command without inputs, Logo displays a Select File dialog with the list of files limited to the ones matching the list of extensions. Logo also supports old-style Terrapin Logo typelists, but their use is not recommended. See `SELECT.FILE` for more information.

LOAD is not available in the trial version or in a published Logo application.

See also `LOADPIC` and `SAVE`.

MAKE

Assigns a value to a name.

Syntax

```
MAKE name object
```

Description

MAKE defines a variable using the name of the first input and assigns the second input as the value of that variable. Once you have created the variable, you may obtain its contents by using :name. Think of the colon (:) as the value of name. To keep a variable local to the procedure in which MAKE is used, see LOCAL or LMAKE.

Remember that all local names of a procedure are visible to any procedure; a MAKE command may, therefore, set a name that is local to a procedure that is calling the current procedure if you use MAKE inside a procedure. See also NAME and THING.

Example

```
MAKE "NUMBER 73
:NUMBER
Result: 73
```

MILLISECONDS

Outputs the number of milliseconds spent.

Syntax

```
MILLISECONDS
```

Description

MILLISECONDS outputs the number of milliseconds spent since the document loaded. MILLISECONDS can be used to track execution times. The accuracy of this timer is supposed to be at least 5 microseconds, but depends on the browser and operating system. Also, it is not guaranteed that the timer started running when the document has been loaded.

NAME

Assigns a value to a name.

Syntax

```
NAME object name
```

Description

NAME defines a variable by assigning the value of the first input to the name of the second input. Once you have created the variable, you may obtain its contents by using :name. Think of the colon (:) as the value of name. NAME is equivalent to MAKE except that inputs are in reverse order. See also LOCAL.

PHELP

Displays help for a property.

Syntax

```
PHELP propertyname
```

Description

PHELP displays the help file for the requested property name.

Example

```
HELP "ALIGN
```

QUOTE

Quotes its input.

Syntax

```
QUOTE word
```

Description

QUOTE quotes its input. Lists are returned unchanged.

Example

```
MAKE "FIRSTNAME "JOHN
:FIRSTNAME
Result: JOHN
QUOTE :FIRSTNAME
Result: "JOHN
```

RESTART

Erases everything and restarts Logo.

Syntax

```
RESTART
(RESTART TRUE)
```

Description

RESTART erases all Logo contents and reinitializes Logo. When RESTART is used, a confirmation dialog box appears. If RESTART and its optional argument TRUE are enclosed in parentheses, Logo will be restarted without the appearance of the confirming dialog!

SAVE

Saves the workspace to disk.

Syntax

```
SAVE filename
SAVE [extensions-list]
(SAVE)
```

Description

SAVE saves the contents of the Logo workspace to a file in the Workspace area. This includes all defined procedures and names, the "PREFS properties, and all buried names. Workspace files are saved as text files.

If no file name extension is specified, SAVE saves the file with the name filename.LGO. To save a file that has no extension, a period is necessary after the filename.

To save the Workspace area, click the menu item “Save Workspace as Workspace.zip”. Note that this command is only available to registered users of Logo.

If you use a list of file extensions as the file name, or use the command without inputs, Logo displays a Select File dialog with the list of files limited to the ones matching the list of extensions. Logo also supports old-style Terrapin Logo typelists, but their use is not recommended. See SELECT.FILE for more information.

SAVE is not available in the trial version or in a published Logo application.

See also LOAD and LOADPIC.

SPLITSCREEN (SS)

Makes both the Output panel and the Graphics canvas visible.

Syntax

```
SPLITSCREEN
```

Description

SPLITSCREEN makes both the Output panel and the Graphics canvas visible.

See also TEXTSCREEN and FULLSCREEN.

Example

```
SS
```

TEXTSCREEN (TS)

Minimizes the Graphics canvas and maximizes the Output panel.

Syntax

```
TEXTSCREEN
```

Description

TEXTSCREEN minimizes the Graphics canvas and maximizes the Output panel.

See also SPLITSCREEN and FULLSCREEN.

Example

```
TS
```

THING

Reports the value of a name.

Syntax

```
THING word
```

Description

THING reports the value associated with the variable named in the input. THING is the Logo primitive that does the same job as : (dots). It can be used to give a variable a second level of evaluation. A variable name could, for example, be passed in to a procedure, and the procedure obtains its value with the THING command. Remember that Logo makes all names in a procedure visible to a procedure that this procedure calls. Therefore, getting a value may result in getting the value of a name that is local to a calling procedure.

Example

```
MAKE "COLOR "BLUE
MAKE "BLUE "AQUAMARINE
THING "COLOR
Result: BLUE
THING :COLOR
Result: AQUAMARINE
THING "BLUE
Result: AQUAMARINE
THING :BLUE
:AQUAMARINE is not a name
```

TIME

Outputs the time.

Syntax

```
TIME
```

Description

TIME reports the current time as a list of three numbers in the form [hour minute second]. The hours are in 24-hour format. See also DATE.

Example

```
TIME
Result: [10 35 50]
```

UNICODE

Converts a character into its Unicode value.

Syntax

```
UNICODE character
ASCII character
```

Description

UNICODE reports the Unicode value of its input. The Unicode standard is a standard code for representing numbers, letters and symbols. If its input is a word, UNICODE reports the UNICODE value of the first character in the word. UNICODE reports an integer value. The input must contain at least one character. The character can be a letter, number or special character. To output the character corresponding to an UNICODE code input, use CHAR.

The alias ASCII is maintained for backwards compatibility.

Example

```
UNICODE "中
Result: 20013
```

VERINFO

Outputs Logo version information as a list.

Syntax

```
VERINFO
```

Description

VERINFO reports information about the version of Logo that is currently running as a list. The elements are major version number, minor version number, subrelease number, the name of the Logo program, the build date as a string, and the name of the operating system.

Example

```
VERINFO
Result: [0 0 1 Terrapin Logo Sat, 01 Aug 2015 08:44:46 GMT Windows]
```

VERSION (VER)

Outputs the Logo version.

Syntax

```
VERSION
```

Description

VERSION reports information about the version of Logo that is currently running.

Example

```
VERSION
Result: Terrapin Logo V0.0.1 Sat, 01 Aug 2015 08:44:46 GMT
```

WHEN

Monitors a change to a Logo property or to a Logo event.

Syntax

```
WHEN [name propertyname compare-to value] [runlist]
WHEN [name propertyname] [runlist]
WHEN [name] [runlist]
WHEN [name propertyname compare-to value] []
WHEN [name propertyname] []
WHEN [name] []
(WHEN [property-condition])
(WHEN)
```

Description

The WHEN command monitors changes to a Logo property, or a change to the Logo environment. Its first input tells WHEN what to watch, and the second input is a list of Logo commands that logo should run when Logo detects the condition to watch,

If the second list is empty, Logo turns off monitoring the the condition described in the first input.

The first input is a list of one to four elements.

If there is only one element, it is the name of the property list to watch; in that case, WHEN triggers the event if **any** of the property list's properties change.

If there are two elements, Logo monitors changes to the supplied property, and runs the runlist if there are any changes to the property. To watch the light sensor of an InO-Bot, for example, the list would be `[INOBOT LIGHT]]`.

If there are three or four elements, the last two elements are a comparison operator (one of =, <>, <=, <, >=, > or any of its

aliases), and the value to compare against. For example, if the APPENDMENUITEM has been used to create a menu item with the ID TEST, the list `[MENU TEST]` would cause Logo to run the runlist whenever the user selects that menu item. For InO-Bot's light sensor to trigger a runlist when the light level is above 0.5, the list would be `[INOBOT LIGHT > 0.5]`.

Note that Logo runs the runlist only once when it compares a value with the <=, <, >=, or > operators. Logo will wait for the comparison to become FALSE before running the runlist again when the condition matches again. This means for the InO-Bot example that Logo would run the runlist once the light level is above 0.5, then wait the light level to drop below 0.5 before checking the level again. This prevents Logo from running the runlist anytime the light level is above 0.5.

Note that you do not use quotes inside a list. For example, if you want Logo to run a runlist when the key “A” is pressed, use `[WHEN KEY = A]`, not `[KEY = “A]`.

You can issue as many WHEN commands as you need to monitor a single property.

A more detailed explanation and many more examples can be found at the page WHEN Something happens.

Note that WHEN stops running when the program terminates. To make Logo monitor changes contiuously even after the program has ended, use the WHENEVER command. This makes, for example, sense when you monitor menu item selections.

Whenever Logo throws a runtime error that is reported to the Listener, Logo removes all monitors.

The WHEN command with a single input is rthe same as the WHEN command with an empty runlist; it stop processing of the condition described in its input.

WHEN without any inputs stops all processing that was created with the WHEN command. It does not stop the processing of runlists defined with the WHENEVER command.

Example

```
; Turn InO-Bot's headlights on when it becomes dark
WHEN [INOBOT LIGHT < 0.3] [PPROP "INOBOT "LIGHTS [1 1]]
; Turn InO-Bot's headlights off when it becomes bright
WHEN [INOBOT LIGHT >= 0.3] [PPROP "INOBOT "LIGHTS [0 0]]
```

WHENEVER

Monitors changes to a Logo property or to a Logo event.

Syntax

```
WHENEVER [name propertyname compare-to value] [runlist]
WHENEVER [name propertyname] [runlist]
WHENEVER [name] [runlist]
WHENEVER [name propertyname compare-to value] []
WHENEVER [name propertyname] []
WHENEVER [name] []
(WHENEVER [property-condition])
(WHENEVER)
```

Description

The WHENEVER command causes Logo to monitor a property of a change to the Logo event. It is identical to the WHEN command, but Logo does **not** stop monitoring when the program that issued the WHENEVER command ends. Logo continues to monitor the condition until a WHENEVER command is entered with the very same condition, but an empty runlist.

Use WHENEVER to monitor changes that need to be monitored outside of a program, like the user selecting a menu item, or a Bluetooth device being connected.

When Logo throws a runtime error that is reported to the Listener, it does not affect runlists defined with the WHENEVER command. Use (WHENEVER) to halt processing for these runlists. It does not stop processing of runlists that were defined with the WHEN command.

Example

```
; Append the menu item "Testing" with the ID "TEST" to the Help menu
APPENDMENUITEM "HELP Testing"TEST
; Print "TESTING anytime the user selects that item
WHENEVER [MENU TEST] [PR "TESTING]
```