Webigami BASIC

For Coders

Welcome to the Webigami BASIC, the Webigami platform's programming language. The lnguage is used in formulas, scripts, integrations and everywhere programming is needed. In fact, it is the only programming language you need to learn. Everything else is configurable point-and-click.

Knowing the language is an important skill since the most advanced Webigami applications require Webigami BASIC. In short, the skill is important since Webigami teams need skilled Webigami programmers.

To become a skilled Webigami programmer you'll need to learn how to move data between variables, cells, sheet storage and external systems. This document discusses that. More specifically, it discusses calculations, triggers, cells, variables, and API elements. These five fundamentals define the language. We start with the rules concerning "calculations."

Note to the reader: Although Webigami BASIC is basic, it is also very specialized. And it's advanced. Give yourself a number of hours to read and re-read this document. That will get you off and running. The language features hundreds of functions so do not expect to learn them in a day. However, references with examples are available. And the language is easy to learn. Good luck.

Calculations

Program snippets in the Webigami world are known as calculations. These programming snippets can be placed in cells and can also be run separately to perform special programming tasks. Each cell's settings section has an option called Cell Calculations. There are also calculation options in the Page Settings area. You can place calculations in either of these two areas.

Calculations can be written two ways: as formulas and as scripts. Formulas are expressions like AB + BC. They are simple and can be easily added to a cell by clicking its settings icon and selecting Cell Calculations.

Formulas can also be written long hand: as scripts. As an example, the formula AB + BC can be written as the following script:

```
MyVar = AB + BC
RETURN MyVar /RETURN
```

Whereas formulas can always be written as scripts, many scripts cannot be written as formulas. Scripts support loops, conditionals, and can be hundreds of lines long. A hypothetical script that calculates a tax based on the rate stored in cell DF and a menu option stored in cell DE appears below.

```
IF DE = "Taxable" THEN
  X = AB * (1 + DF)
ELSE
  X = AB
/IF
RETURN X /RETURN
```

The option to write a calculation as a formula is available only when you're writing to the current cell. Everywhere else you use scripts.

In summary, calculations can be placed in cells and in your Page Settings area. They sometimes can be written as formulas and always as scripts. How calculations are triggered and used is the topic of the next section.

Triggers

The word "trigger" means "under what circumstances a calculation runs." Webigami sheets support a wide variety of triggers. Calculations that run when a sheet is recalculated are called "spreadsheet-like." They are extremely usefull, hence the popularity of commercial spreadsheet programs.

Despite their power, spreadsheet-like calculations are insufficient for building advanced applications. You'll want to run calculations in response to cells being edited by end-users, when Action buttons are clicked, and when users first enter a sheet. Triggers provide you that ability. The triggers below are available.

Trigger	**Run calculation when...**
Spreadsheet	Runs when sheet is recalculated.
OnEdit	After a cell is edited.
OnAppEntry	When an application is first loaded.
OnDesignEntry	After a sheet's data is first loaded.
OnClick	When a Go button or Action button is clicked
OnActionBefore	When a built-in Action button is clicked
OnActionAfter	After a built-in Action button completes its tasks.

The first two items above relate to specific cells. These scripts run when the sheet is recalculated or when the cell is edited, but not both.

The bottom set are sheet-related. Set them up in the Extensions - Calculations area. Place the functions you define in the OnAppEntry calculation.

In summary, calculations can be placed in cells and in sheets. The calculations in cells run when the sheet is recalculated or when the cell is edited. Calculations in sheets are used for running scripts when pages are loaded, saved and when buttons are selected. In the next section we look at the nature of cells.

Webigami Cells

Webigami sheets contain different kinds of cells. A cell can be simple or complex. It can be spreadsheet-like or triggered. It can be a header cell or a table cell. We discuss the different kinds of cells below. We also discuss how these cells fit into the world of applications, where sets of sheets are combined together to manage multiple work routines.

Simple and Complex Cells
Cells are either simple or complex. Simple cells store a single value. Complex cells store a table of values. Examples of simple cells include number and text boxes. Examples of complex cells include File Attachments, Agendas, and Scribbles (cells that store drawings and signatures).

Complex cells require you know how the table of values is structured. You can find a cells structure with the INFO function. Specify the cell you want to learn about to understand how to read and write its values. For example:

```
INFO("AB")
```

Header and Table Cells
Cells can appear inside and outside the cell tables on a sheet. The cells sitting outside cell tables are called "header" cells. They're stand-alone. Each has its own two-character cell code for use in formulas and scripts. Cells inside cell tables are called "table" cells. Table cells are organized into columns. Each cell in a column has the same two-character cell code, the same (repeated) script, and each cell in the column uses the same kind of cell component (for example, the same text box or the same menu selection).

You refer to an entire column's data wth the cell code by itself. You refer to the value of specific cell in a column with the cell code and its row number. See the two examples below.

X=FA	Place all values from column FA into X
X=FA[2]	Place the value from column FA, row 2 into X

Spreadsheet-like Vs. Triggered Cells
Cell calculations can be spreadsheet-like or triggered but not both. Spreadsheet-like calculations write to themselves and cannot write to other cells. They run when the sheet is being recalculated. The value they return becomes the value of the cell. Spreadsheet-like scripts use RETURN statements to return their calculated results and end execution.

Triggered calculations work the opposite of spreadsheet-like calculations. They can write to other cells but they cannot write to themselves. They do not return any data so they do not use RETURN statements. Instead, triggered scripts end when they encounter an EXIT or the end of the script is reached.

Other than these noted differences, spreadsheet-like and triggered calculations run in the same way. Both use the same Webigami BASIC language.

Reading and Writing to Sheets in an Application
In the Webigami world, applications are sets of designs. Together, they form an application. As an application, the calculations are allowed to read and write to other sheets in the same application. They cannot read or write to sheets in other applications.

Also, as a user moves between sheets in an application, the cells changed on one sheet are automatically available to the next. In other words, they're "global" to the application. These cells are referenced by their cell codes and design IDs. For example: AB!1000. (The design ID in this example is 1000).

As mentioned above, cell codes are global. To be more specific, they're global variables. Global variables are discussed in greater detail in the next section.

> Loading And Saving Sheets
> Loading and saving data in sheets is typically left up to the Webigami environment. The user enters a sheet and data is loaded from permanent storage. The user clicks Save and data is saved back to permanent storage.
>
> However, when building multi-sheet applications, you often want to take control of the loading and saving of data. You want to set up "Non-storage" sheets that load and save data to and from other sheets. In these cases you load and store data with scripts. For information on this topic see the documents on application development.

To summarize this section, we find that cells can be header and table cells. They can hold a single value and they can be complex. They can be spreadsheet-like and run when the sheet is recalculated, or they can be triggered when a user edits the cell. Also, scripts can read and write to other cells. And when the sheet is part of an application, the cells across all the sheets become available to the scripts in the sheets.

In the next section we look at a larger topic: the topic of variables. You'll find cells are just a kind of variable.

Variables

In this section we list five rules governing variables in Webigami BASIC. We learn more about the syntax of the language, naming and formatting conventions, and the difference between local and global variables.

RULE #1
Variables in Webigami BASIC are two-dimensional. In other words, variables are tables of values. Even when you write X=1 you're creating a table-- it just happens to have one column and one row with the value "1" in it.

You address Webigami variables as follows:

• To address the entire table of values specify the variable name. For example: **MyVar**
• To address column i, row j of a table follow the name with the column and row in brackets. For example: **MyVar[2,3**] or **MyVar[i,j]** *Note: Spaces are not allowed between the name and the left square bracket.*
• To address the i'th row of column 1 follow the name with the row number in brackets. For example: **MyVar[2]** or **MyVar[i]**. *Note: **MyVar[1,2]** is the same as **myVar[2]**.*

Use the same conventions above when addressing Webigami cells. For example, AB[2] and AB!123.1000[2] indicate the cell in the second row of column AB on the current design and in the design 123.1000, respectively.

Global Variables
Webigami variables can be local and global. Local variables are available (persistent) during execution of the script and then dropped. Global variables keep their information as long as you stay in the application, within the designs that make up the application. You decide whether a variable is local or global by the name you give it. See below.

	Local Variables	Global Variables
Explanation	Use standard variable name.	Standard name plus a "!" or a "!" and a design ID.
Example	X	X!
Example	X[1,2]	X![1,2]
Example	AB	AB!1000
Example	AB[4]	AB!1000[4]

Note: The value of an undefined value (its default) is ERROR (with one row and one column). The default value of a value other than the one in column one, row one is the empty string. Webigami BASIC does not have the concept of a NULL value. To clear a variable you set it to ERROR.

RULE #2
Variables are auto-datatyped. In other words, you do not need to declare whether a value is a date or a number and how many columns and rows you want. You just start using them.

Values can be text, numbers, currencies, dates and times, and logicals. Make sure to write them properly. Numbers cannot contain commas and special characters. Dates and times must be written as MM/DD/YYYY, or HH:MM, or MM/DD/YYYY HH:MM where M means month, D means day, Y means year, H means hour and M means minute (respectively). Logicals return the value TRUE or FALSE. for example, IF A="TRUE" then ... /IF.

RULE #3
Variable names can be upper or lower case. In other words, the variable ABC and the variable Abc are the same. Also variable names cannot begin with a number and cannot contain special characters like stars and quoting characters.

RULE #4
To specify a literal value place quotes around it (quotes around numbers are optional). Webigami BASIC supports three types of quoting characters: single quotes, double quotes, and carets. When quoting a value you must begin and end with the same quoting character. The values in quotes may contain the other kinds of quoting characters but not the one used. The examples below demonstrate the use of different quoting characters.

```
MyMsg1 = ^He asked, "Was that Jessy's first time?"^
MyMsg2 = "Your password is: 3a^%_1234"
MyMsg3 = 'Do not use quotes (") in your password.'
```

RULE #5
The four basic arithmetic operators are +, -, *, and /. These do addition, subtraction, multiplication and division respectively. For example:

```
X = (AB * MyFactor - 3)/C
```

The "test" operators used in statements like "IF A > B THEN ... /IF" are as follows:

=	Equals.
!=	Not equals.
=	Equals
==	Equals (Case-sensitive)
<	Less than.
<=	Less than or equals.
>	Greater than.
>=	Greater than or equals.
~	Text begins with.
~~	Text contains.

The text-concatenation operator is the & character. It's used to paste text together. Technically speaking, you can also use a + character, but if the values being pasted together are numbers the language will add the number instead of pasting them. It's best to use the & character when pasting text together and the + character when adding numbers.

Example	Result
X="ABC" Y=100 RETURN X&Y /RETURN	ABC100

Note: Additional testing operations are available as API Functions.

RULE #6
Line-item cells run their scripts across all their lines. In other words, the same script runs repeatedly, once for every line. Use the functions *ThisRow*, *ThisCC*, *IsSummary(CC)*, and *Summary(CC)* to get the current row number, the current cell code, whether the cell has a summary row, and the value in the summary row, respectively. See examples below.

Line Item Calculation Examples
X = AB[thisRow] + AC[thisRow] RETURN X /RETURN
Y = AB[thisRow] + Summary("AC") RETURN Y /RETURN
Y = AB[thisRow] + Summary(ThisCC) RETURN Y /RETURN
Z = LastRow RETURN Z /RETURN

IF ISSUMMARY(ThisCC)="TRUE" THEN
RETURN SUMMARY(ThisCC) /RETURN
ELSE
RETURN "" /RETURN
/IF

In summary, Webigami BASIC variables are two-dimensional, auto-datatyped, case-insensitive, and can be local and global. To make a variable global you add a "!" to the end of it (Example: myVar!). To address a specific column and row you can specify just the row (Example: MyVar[2]) which assumes column 1, or you can specify both the column and the row (Example: MyVar[1,2]).

API Elements (Functions and Statements)

Webigami BASIC includes hundreds of functions. To access them use the link provided wherever script boxes appear. Your language reference has the six categories below.

System Functions - These functions provide information about your environment: the login ID of the user, the last design they visited, whether they're a guest or staff, the current date, and so on.

Loops and Conditionals - FOR loops, IF statements, and functions that tell you whether a logical condition is TRUE or FALSE.

Dates, Numbers and Text - A wide variety of functions that manipulate dates, numbers and text.

Data Storage and Search - Functions that retrieve, store and search data stored on Webigami sheets.

2D Array Manipulation - A wide variety of functions for manipulating variables with multiple rows and/or columns.

Specialty Cell APIs - Functions that update, read and write to complex cells like File Attachments, Agendas, and Scribbles.

See your on-line reference to view the functions available.

Next Steps

Webigami BASIC provides a complete programming environment for turning Webigami sheets into full-featured applications. Automating the most complex work routines is about designing the sheets you need, setting them up, and adding code to them.

As a next step, dig in. Pick a programming tutorial on an application of interest to you. Then follow it. If you get stuck, reference other tutorials, the on-line programming reference, videos or leave Webigami a message.

This document was written by Dave. Have ideas on how to improve this article? Share them with us. We're always trying to improve.