# WebWorksheet ™

Version 3.6

# User Manual

# Contents

# Installing WebWorksheet

WebWorksheet is comprised of two software components:  the Microsoft Excel add-in that is installed on the licensed user's workstation, and the shared component which is referenced from within the generated web page.  Only the Microsoft Excel add-in requires installation.

Follow these steps to install the add-in on your workstation:

1. Download the self-extracting executable using the instructions provided in the confirmation email you received after the purchase was completed.  This file can be saved anywhere on your computer.

2. Extract the WebWorksheet files by double-clicking on the downloaded file.  Files will be installed in the C:\Program Files\Webworksheet folder.

   **Windows Vista, Windows 7, and Windows 8 users:**  You must run the installer program as Administrator using the right click menu.
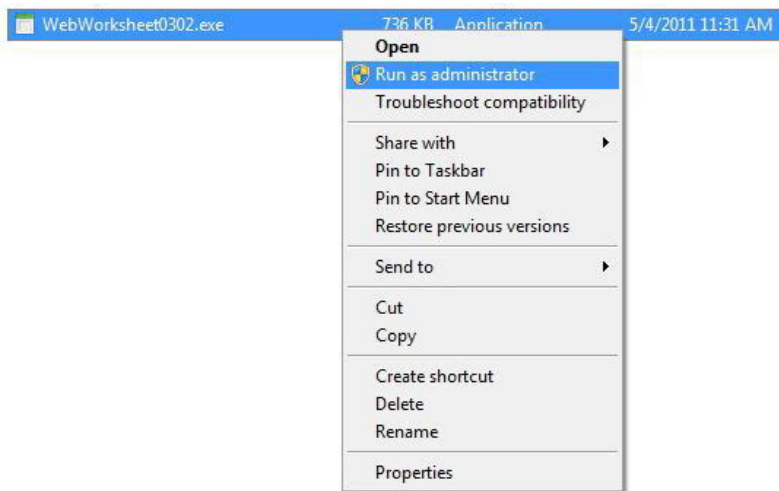
   

3. Once the software has been installed, the add-in must be enabled in Microsoft Excel.

   In Microsoft Office Excel 2003, select the Add-Ins... option on the Tools menu.
   Then using the Browse button, select the webworksheet.xla file from the C:\Program Files\Webworksheet folder and click OK.  The WebWorksheet toolbar will now be visible.  This toolbar will automatically appear each time Microsoft Excel is started.  To temporarily disable the WebWorksheet toolbar, uncheck the WebWorksheet option on the Add-Ins menu, and the toolbar will not be visible.  Simply re-check the WebWorksheet option to make it visible again.
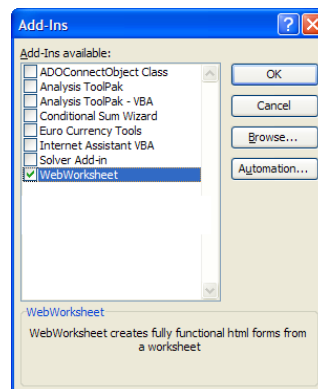
4. In Microsoft Excel 2007 and later, click the Office button, choose Excel Options, select the Add-Ins screen, choose Excel Add-Ins in the Manage input box and click Go. In the Add-Ins dialog, click the Browse button. This will open a familiar Open File dialog. Navigate to the C:\Program Files\Webworksheet folder and select webworksheet.xla, and click Open.

5. When installation is complete, the C:\Program Files\Webworksheet folder will contain the Excel add-in file, the user manual (this document), and several example spreadsheets which demonstrate the features of WebWorksheet.

# Creating a Web-Enabled Spreadsheet

Once the Microsoft Excel add-in is installed, open the workbook containing the sheet you would like to publish on the web. WebWorksheet publishes each worksheet as a separate file, so if there are multiple worksheets in the same workbook to be published, follow this process for each worksheet.

## *Creating the WebWorksheet*

When converting an existing spreadsheet to a webworksheet, we suggest creating a copy of the worksheet inside the workbook. This is done to keep the original worksheet intact for later use. Rename the copy of the worksheet to something meaningful, as the worksheet name is used for both the name of the html file that is created and for the title which appears on the browser tab when this worksheet is viewed on the web.

For example, if the original worksheet which contains the timesheet is named "Sheet 1", create a copy and rename it to "Weekly Timesheet". WebWorksheet will create a file called Weekly_Timesheet.htm to publish on the web, and when viewed online, it will appear with the worksheet name as the browser tab name.



Worksheet Tab Names                      Browser Tab Names

> *Tip*: Avoid using special characters in the worksheet name as some of those characters cannot be used in a filename, and WebWorksheet will replace them with the underscore character.  WebWorksheet will also replace all spaces in the filename with underscores, but the spaces will remain in the tab name.

From this point forward, any reference to a worksheet refers **to the copy** of the original worksheet.  The original worksheet should not be modified during the process of creating a web-enabled worksheet.

Identify the bottom-right cell of the worksheet and place the "#end" marker (without quotes) in that cell.  This identifies to WebWorksheet the ending row and column to be included in the web-enabled version.  Any cells, including data lookup cells referenced using vlookup or hlookup, must be inside the marker.  The row and column which contains the marker is **not** included in the html version, so place it one column outside and one row below the content to be published.

If the '#end' marker is not found in the worksheet, the following error will be displayed:
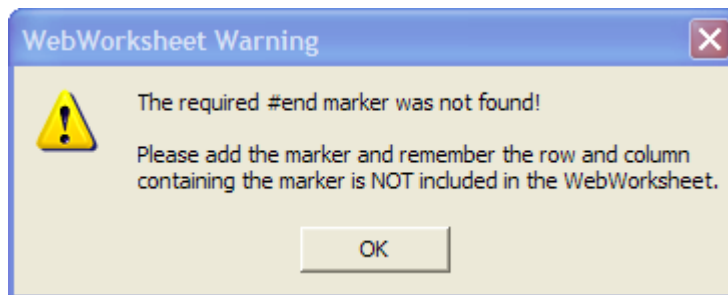


The maximum size of the WebWorksheet is 1000 rows and 256 columns (A1:IV1000).

> *Tip*: To make data lookup cells invisible in the .htm version, simply hide those rows or columns in the worksheet, but keep them inside the #end marker.

> *Tip*: If you use ranges to define the values for your dropdown cells, place them in rows at the bottom of your worksheet. The #end marker can then be placed above those rows.  The dropdown values are needed only when the HTML page is generated, not at run time, so they do not need to be included (or hidden) in the generated page.  This will make the generated page smaller and run faster.

At this point, it is suggested that you create the html page and view it so any formatting differences can be resolved.  To create the webworksheet, select the "Create a WebWorksheet" command from the WebWorksheet toolbar.
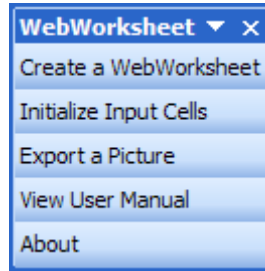
A file will be created in the same folder as the worksheet, and will be named according to the worksheet name (e.g. Weekly_Timesheet.htm).  To view the generated file, simple double-click on its name, and the .htm file will be loaded into the browser defined as the default for your workstation.  The Excel worksheet and the web page can now be viewed side by side for comparison.

## *Formatting the WebWorksheet*

Correcting any format differences usually entails setting the cell format appropriately.  Excel is very forgiving in certain ways, and makes assumptions as to how to display the information you entered.  Sometimes those assumptions do not translate well to the web, so Excel must be explicitly told how to display the information.  You will need to verify each cell is displayed as you prefer on the webworksheet.

**Borders**:  Verify the cell borders are the correct color, thickness, and style (e.g. solid or dashed).  Oftentimes in a worksheet we rely on the gridlines to provide the visible borders, but gridlines are not displayed on the web version.  Use the Format…Cells Border tab to change the borders for a cell.

> *Tip*: Include a thin, blank column on the left and right of your form.  This will help to clearly see how the left and right borders are being formatted.

**Font Style, Size, and Color**: If necessary, the Format…Cells Font tab should be used to set the desired font family (e.g. Arial or Verdana), size, and color.  Most cells default their color to "Automatic", which is translated to black by WebWorksheet, so it is not necessary to force those cells to be black.

**Merging Cells**:  If any text appears cut-off or missing on the webworksheet, it's probably because it does not fit in the cell with its current settings for font style or size.  Again, Excel is forgiving in this regard, and will show the text if the adjacent cell is empty.  The web cannot do that.  The easiest solution is to merge adjacent cells ( both horizontally and vertically ) to accommodate the text.

As illustration, the following 3x3 section of a worksheet will allow the text in cell A2 to be shown it its entirety.



When this same worksheet is converted to html, the resulting page looks as follows:

| | | |
|---|---|---|
| this text | | |
| | | |

Merging cells A2, B2, and C2, and then recreating the webworksheet, will yield:

| | | |
|---|---|---|
| this text is larger than the cell | | |
| | | |

**Cell Alignment**: Excel oftentimes makes assumptions on whether the text in a cell should be left-aligned or right-aligned depending on the type of data in the cell ( e.g. a date, a number, a text string ).  To create a professional looking web page, you may want to force the alignment by using the Format...Cells Alignment tab.  You may also want to set the Indent on a cell to give it a fixed margin on the left or right sides so the text does not touch the cell border.

As an illustration of the advantage of using indentation, consider the following section of a worksheet:

| | A | B |
|---|---|---|
| 1 | | 3 |
| 2 | | 4 |
| 3 | Total | 7 |

When converted to html, the following is displayed in the browser:

| | |
|---|---|
| | 3 |
| | 4 |
| Total | 7 |

To give a little separation between the text and borders, we can set up Column A to have a right indent of 1, and Column B to have a left indent of 1, yielding the following:

| | |
|---|---|
| | 3 |
| | 4 |
| Total | 7 |

Cell vertical alignment ( top, center, or bottom ) should also be reviewed and adjusted to improve the look of the web form.

The steps to correct any formatting differences can be repeated as often as necessary until the webworksheet is an identical replica of the Excel worksheet.

## Identifying Input Cells

The next step is to identify each input cell for which the user is to enter a value.  When using an Excel worksheet, every cell is available for input, but that is probably not the best

approach for deploying a web form. WebWorksheet provides several functions for collecting input from the user, so choose the method which best meets your needs.

- Use the wwsInput() function to collect information to be typed by the user, including numbers, dates, and text strings, or

- Use the wwsDropDown() function to create a list of options for the user, and they select one of those options from the list, or

- Use the wwsCheckBox() function to create a checkbox which the user can check or uncheck. Multiple checkboxes can be grouped together so the user can select only a single option from a list of multiple options.

- Use the wwsCalendar() function to create a popup date-picker (calendar) to allow the user to select a date via a click. The calendar can be configured to appear automatically when the cell is activated or when a calendar icon is clicked.

For example, to identify input cells for the upper portion of the Weekly Timesheet, we would enter =wwsInput() into each of the appropriate cells:



If a default value is desired, that value can be passed to the wwsInput function, as shown in the Manager field above. That default value will be displayed in the input cell but may be deleted or changed by the user.

See the section on Input Functions for a detailed description of each input function, its arguments, and more examples.

To expedite the setup of input cells, you can use the Initialize Input Cells command on the WebWorksheet toolbar. Select one or more cells (cells do not have to be contiguous), then select the command on the toolbar. The following message will appear:
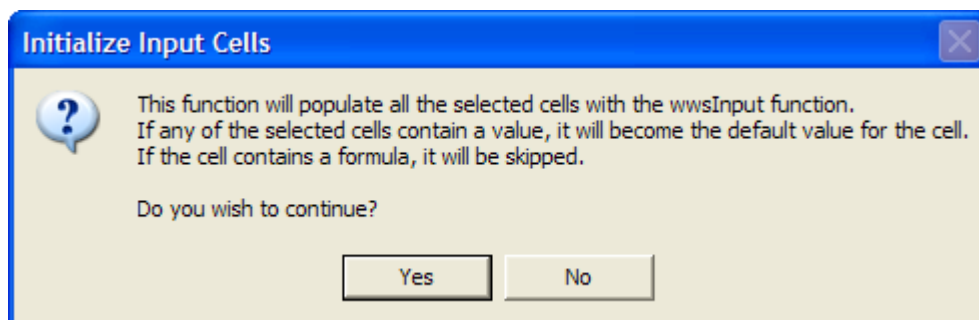
If Yes is selected, each of the selected cells which do not contain a formula will be given the formula =wwsInput().  If the cell contains a value, it will become the default value for the wwsInput function.  For example, if the cell contained the value "<Enter your name here>", it will be given the formula =wwsInput("<Enter your name here>").  If the cell contains a formula, a message will be displayed containing the cell number and its formula:
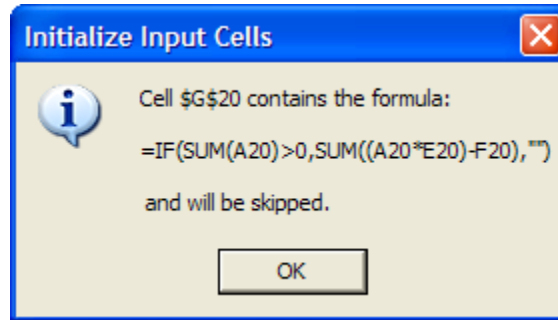


Cells which require input via the wwsCheckbox, wwsDropdown, or wwsCalendar functions must be manually defined.

## Validating User Input

If desired, cell validation rules may be defined for input cells.  Validation rules can be used for prompting users when they arrive at an input cell, verifying the data entered meets certain criteria, or preventing a user from submitting (emailing) a webworksheet with missing information.

All validation rules are defined using Excel validation criteria, which are found under the Data…Validation menu.  While it is outside the intent of this user manual to describe all the options of using Excel validation, the following describes some of the key features.  More information on validation rules can be found at http://support.microsoft.com/kb/211485. See Appendix B for additional information on validating checkboxes before submitting a form.

To prompt the user when an input cell is selected, use the Input Message tab and check the Show Input … box to enter the desired Title and Input Message.  Following is an example input message defined for the Weekly Timesheet Employee name field:
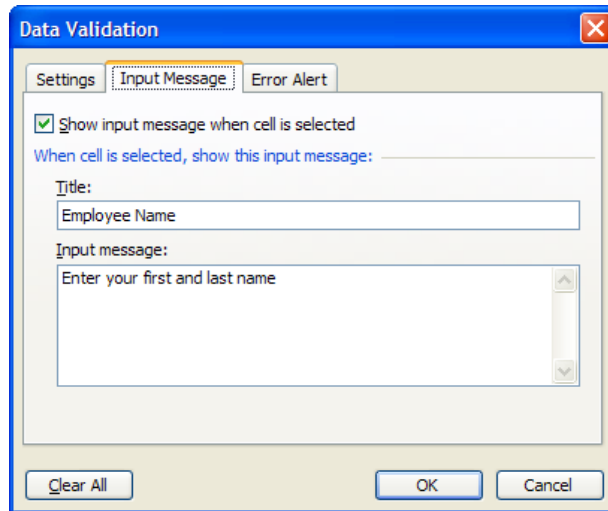
When the Input Message is defined, Excel will show the prompt whenever that cell is active.



When the WebWorksheet is created, that same input message will be displayed as:



To validate the entry made by the user conforms to some criteria, rules are established on the Settings tab.  Excel provides the ability to validate an entry as a whole number, a decimal number, a date, a time, a certain length, or a member of a list of values.  It can also be used to verify a value or length is a fixed value, greater than or less than a value, or somewhere in between.  Custom rules can also be defined.

The Error Alert tab is used to define the message to display when cell validation fails and to define how that error affects form submission. Excel provides three levels of alerts: Stop, Warning, and Information.

When the Style is set to Stop, WebWorksheet will require the user input to be present and pass the validation rule defined on the Settings tab before the webworksheet can be submitted (emailed). When the submit button is clicked, WebWorksheet will check all the values, and if missing or fails the validation rule, a message will be displayed to the user and the errant fields will be highlighted in red.

For example, assume the Employee name is mandatory. The Settings tab would be used to define a minimum length for the name, such as:
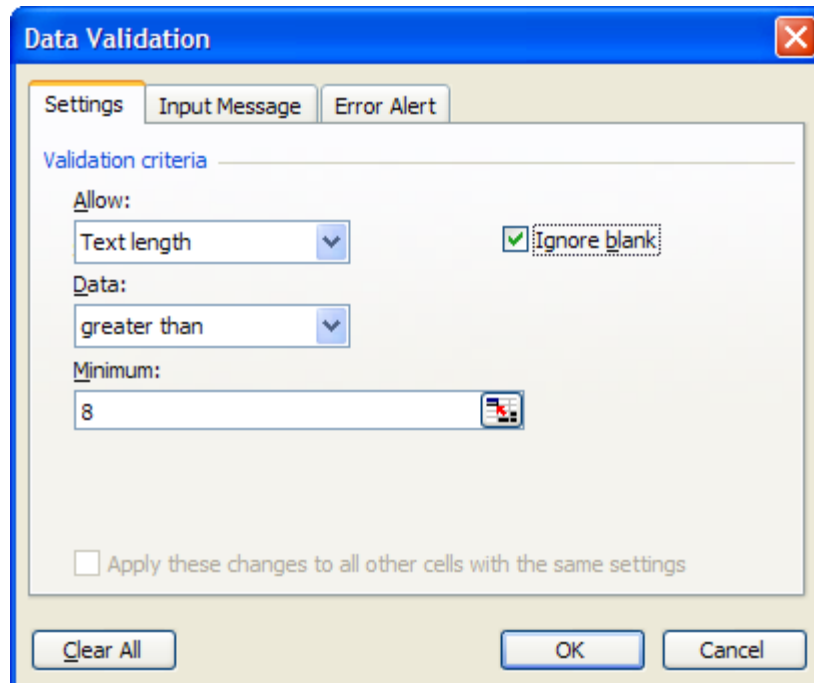
Then the Error Alert tab would be set up as:
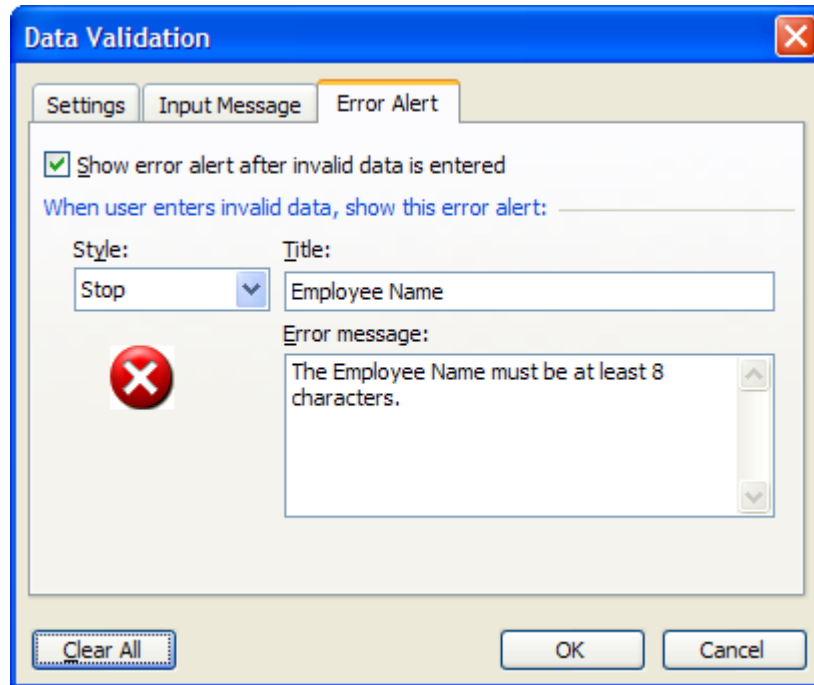


If the user selected the submit button with the employee name missing, WebWorksheet would display the following:

If the error style is set to Warning, the offending fields will be highlighted in yellow and the following message would be displayed:

| [Street Address] | | Employee: | |
| [Address 2] | | Manager: | |
| [City, ST ZIP Code] | | Employee phone: | |
| | | Employee e-mail: | |
| Week ending: | | | |

| Day | Regular Hours | Overtime | Sick | Vacation | Total |
|---|---|---|---|---|---|
| Monday | | | | | 0 |
| Tuesday | | | | | 0 |
| Wednesday | | | | | 0 |
| Thursday | | | | | 0 |
| Friday | | | | | 0 |
| Saturday | | | | | 0 |
| Sunday | | | | | 0 |
| Total hours | | | | | |
| Rate per hour | | | | | |
| Total pay | $0.00 | $0.00 | $0.00 | $0.00 | $0.00 |

**Message from webpage**

? 1 recommended field(s) are missing and have been highlighted in yellow.

The employee name must be at least 8 characters.

Do you wish to submit the form with missing fields?

OK      Cancel

The user may choose to submit the form with the missing fields (via the OK button), or Cancel to correct those fields. If the Style is set to Information, no validation occurs and the submit will be executed.

## *Defining Mouseover Effects*

A mouseover can be defined for any non-input cell on the webworksheet by defining a comment for that cell in Excel. Using the Insert…Comment menu option, define the message to be displayed on the webworksheet whenever the user hovers the mouse over that field. The red indicator that Excel uses to identify cells with comments is <u>not</u> visible on the webworksheet.

For example, to define a mouseover for the Overtime column header, define a comment in Excel as:

| Hours | Overtime | | cation | Total |
|---|---|---|---|---|

Overtime is defined as any time over 8 hours in a given day or any hours on a weekend or holiday.

When the mouse hovers over the Overtime header on the webworksheet, the following will be displayed:

| Hours | Overtime | Sick | Vacation | Total |
|---|---|---|---|---|
|  | | | | |
|  | | | | |

Overtime is defined as any time over 8 hours in a given day or any hours on a weekend or holiday.

*Tip*: To set up mouseovers for input cells, use the Input Message tab on the cell validation rules.

### Highlighting the Active Cell

By default, Microsoft Excel places a thick, black border around the active cell (the cell currently selected).  While this works well within Excel, it may not be the ideal or desired way to identify the active input cell on the web.  Therefore, WebWorksheet provides functions for you to customize the active cell.

Use the wwsActiveBorder() function to define how the border around the cell should be formatted, including no border at all.  The wwsActiveBackground() function can be used to define the background color of the active cell, or transparent to allow the color of the cell to show through.  If no border or background is defined, it will default to the standard Excel black border.

### Including Images in your WebWorksheet

Microsoft Excel workbooks may contain images, such as corporate logos or product pictures, which you may want included in the generated webworksheet.  Since images are oftentimes not linked to a specific cell, it is not possible to automatically extract those images for placement on the webworksheet.  Therefore, we provide the Export a Picture option on the WebWorksheet toolbar.

Each image must be extracted individually the first time.  Once each image is extracted, it may be used in multiple webworksheets or multiple times within the same webworksheet.  To extract the image, select it, then click on Export a Picture.

WebWorksheet ▼ ✕
Create a WebWorksheet
Initialize Input Cells
Export a Picture
View User Man  Save an embedded image as a file
About
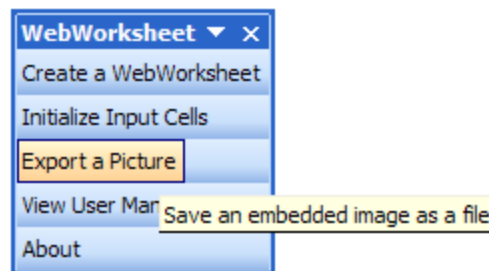
If the command is selected without a selected image, the following message will appear:

WebWorksheet                                    ✕

A picture must be selected to export.

OK
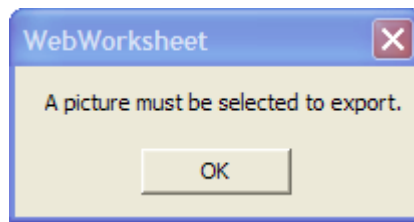
If an image is selected, you will be prompted to enter a filename to store the extracted image:

WebWorksheet                                    ✕

What would you like to name this image?         OK

Cancel
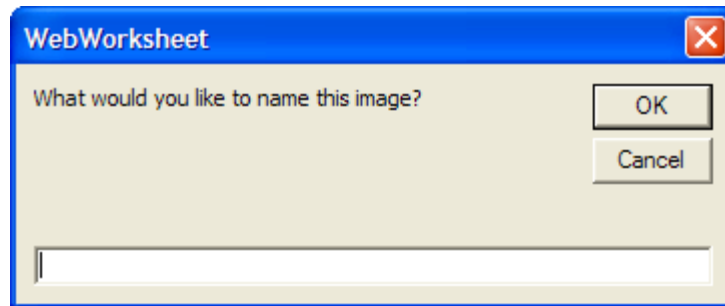
Enter a descriptive name and then click OK.  The image will be copied as a .jpg file and stored in a subfolder called 'wwsImages' under the same folder as the Excel worksheet.  WebWorksheet will confirm the export and give the size of the image, in pixels, for use in the wwsImage() function.  The original image is left intact on the worksheet.  Repeat this process for each image.

WebWorksheet                                    ✕

The selected image has been exported to C:\Inetpub\wwwroot\webworksheet\test.jpg with a height of 64 pixels and a width of 72 pixels.

OK
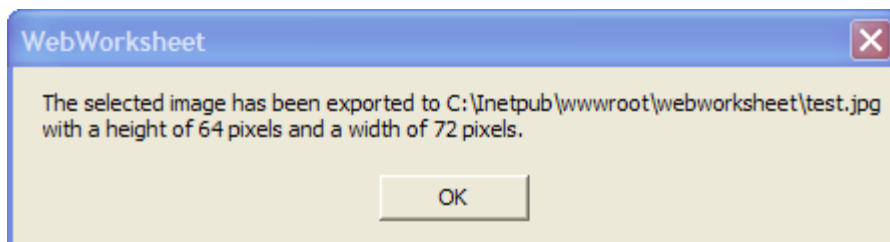
To place the image on the webworksheet, use the wwsImage() function in a cell.

Using the Excel Format...Sheet Background menu, it is possible to define a background image for the worksheet.  The image will be repeated across the page behind the cells.  To replicate this feature on a webworksheet, the wwsBackground() function is used.  Place this formula in any cell on the worksheet.  The function returns the name of the image but that name does not appear on the webworksheet.

For example, placing the formula =wwsBackground("background.jpg") in cell B1

| | B | C | D | E | F |
|---|---|---|---|---|---|
| 1 | background.jpg | | | | |
| 2 | [Company Name] | | | | |
| 3 | | | | | |
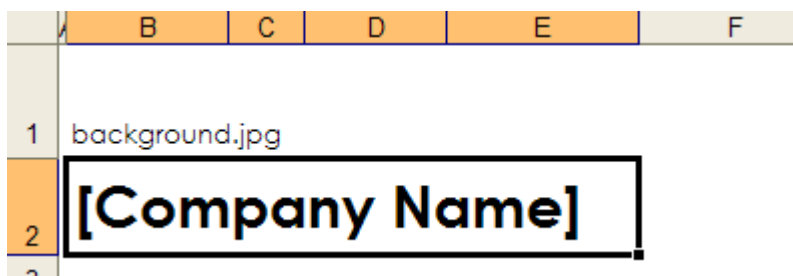
would result in the following webworksheet:



Microsoft Excel also supports images within cell comments, which are displayed whenever the mouse hovers over the cell. WebWorksheet will automatically export those images and place them in the 'wwsImages' subfolder, and those images will be displayed on the webworksheet form when the mouse hovers over the cell. Each of these images will be named as <tab>_<cell>.jpg, where <tab> is the name of the worksheet tab and <cell> is the cell range (e.g. B4).

> *Tip*: Remember to copy the entire 'wwsImage' subfolder to your website when publishing the WebWorksheet.

## *Creating Intelligent Forms*

While all of the techniques described above will allow you to create a fully-functional spreadsheet for the web, we've also created a few functions which can be used to make the user experience even more satisfying.

One technique widely used, particularly on large complex forms, is to show and hide sections of the form based on user input. WebWorksheet provides several functions for controlling parts of the form, so choose the method which best meets your needs.

- Use the wwsToggle() function to show and hide a specific number of rows immediately following the row containing this function. This is ideal for FAQ sections or including the symbols for expanding [+] or collapsing [-] sections of a form.

- Use the wwsShowRows() and wwsHideRows() functions to show or hide specific rows on the form, which can be triggered via a calculation, or use wwsShowAndHide() to both show and hide rows based on user action. The wwsVisible() function can be used to check if a row is visible or hidden.

- Use the wwsBookmark() function to create a location marker inside your form, which when used with the hyperlink function inherent in Microsoft Excel, can position the user at any specific point on the webworksheet.

## *Processing  Completed Forms*

One of the most powerful features of WebWorksheet is the ability to have completed forms emailed to an address when the user has completed data entry.  The wwsSubmitButton() function allows the form to be sent via email to a defined address.  The completed form is contained within the body of the email message, and can also be included as a file attachment for easy archival by the recipient.  All the validation techniques described above can be used to ensure the form is completed properly before allowing it to be sent.  As a security feature, WebWorksheets which are sent via email cannot be modified by the recipient.

The wwsClearButton() function removes any user entry from the webworksheet and restores all the fields to their original values.

Forms can also be saved on the local workstation before being emailed, so the user can complete sections of the form and return later for more data entry.  The wwsSaveButton() function allows the webworksheet to be saved with all user entry intact.  To complete entry, the user selects the local copy and can save as many times as desired.  When complete, the form can be sent via email if the submit button has been defined.

The wwsPrintButton() function allows the completed form to be printed using the standard Windows Print Dialog box.  Optionally, a message can be displayed to the user before the Print Dialog box appears as a reminder to set specific printing options, such as landscape.

Prior to emailing, saving, or printing your form, it is also possible to hide or show specific rows, or to change the value of a specific cell.  The wwsUserClicked() and wwsSetCell() functions are used to execute functions as a result of a button click.  For example, you may wish to hide some user instructions before emailing the form, or to make sure the dynamically hidden rows are visible on the emailed version.

## *Protecting Your Information*

Your business information is important, and you may not want your web users to see the data or calculations embedded in your webworksheet.  That information may be protected using the wwsProtectPage() function, which encrypts the information using highly secure algorithms.  When a page is protected, the user may be optionally prompted to enter a password before the webworksheet will be displayed.  If the user attempts to view the embedded data using the view..source command on the browser, they will see just a jumbled array of numbers and letters.

## *Deploying Your WebWorksheet to the Web*

Once you have completed the webworksheet, follow these steps to deploy it to your website.  You may need assistance from your IT staff or hosting provider to move the files to their proper destination and configure access.

Only the .htm and .jpg files created by WebWorksheet need to be copied to the web.  Other files used by WebWorksheet to provide the computational power and page formats are provided on WebWorksheet's website, and your pages reference those.  The Microsoft Excel add-in file also does not need to be deployed.

To deploy a webworksheet, copy the .htm file created by WebWorksheet to the folder within your website which contains all the other pages for your website.  The .htm file is named after the tab name defined in Excel (e.g. if your worksheet tab was named Timesheet, the file created by

WebWorksheet will be named Timesheet.htm), so naming conflicts should not exist.  If they do, rename the tab in Excel and regenerate the .htm file.

If your worksheet contains any images, the wwsImages folder and all its contents must also be copied to the website to the same location as the .htm file.  If you exported any images using the toolbar function, or if you have images embedded in the cell comments, then WebWorksheet automatically placed .jpg files into the wwsImages folder.

Lastly, because your .htm pages reference files which reside on the WebWorksheet website, your website must allow access to http://webworksheet.com/release/*.  Typically, there are no changes necessary to allow this access, but some organizations have very strict security profiles in place.  If your security profiles do not allow access to external sites, please contact us regarding alternative forms of deployment.

# Input Functions

## wwsInput

### Description

Accepts keyboard input for a cell.  May be used for dates, numbers, or text strings.

### Arguments

*default_value* (optional) – The initial value of the cell, which may be overwritten by the user.  If no default value is provided, the cell is empty.  The initial value may be an integer, string, date, formula, or cell reference.

*showAsPassword*(optional) – When set to TRUE, the input box will show only asterisks for each letter typed. If FALSE, or not provided, the typed letters will be shown.

### Validation Options

User input may be validated to be a specific value, within a range of values, or of a specific type.  Standard Microsoft Excel validation functions are used to define the validations (see http://support.microsoft.com/kb/211485).

### Examples

=wwsInput()

=wwsInput(2.5)

=wwsInput("<enter your name here>")

=wwsInput("12/25/2009")

=wwsInput(B3+29)

=wwsInput(TODAY())

=wwsInput(15%)

=wwsInput("", TRUE)

---

*Tip*: Set up the validation rules for the cell prior to entering the wwsInput function, otherwise Excel will complain that the formula does not pass the validation rules.

---

*Tip*: To create multi-line input boxes, set the height and width of the cell(s) to the desired size and set the Alignment to Wrap text on the Format Cells menu.

---

## wwsDropDown

### Description

Places a dropdown list in the cell from which the user may select a single option.

### Arguments

*option_values* (optional) – The comma-delimited string or range which contains the option values.  If no string or range is provided, the list settings in the validation rules will be used.  If no option values are defined, an error message will be displayed.

### Validation Options

The options in the list may be defined using standard Microsoft Excel list validation functions (see http://support.microsoft.com/kb/211485).  When using the validation rules to define the list of options, the list can be specified as either a cell range or a comma-delimeted list.  See Appendix C for details on creating dependent dropdowns.

### Examples

=wwsDropDown("Yes,No,Maybe")

=wwsDropDown(A1:B5)

=wwsDropDown()  *with range defined in validation rule*
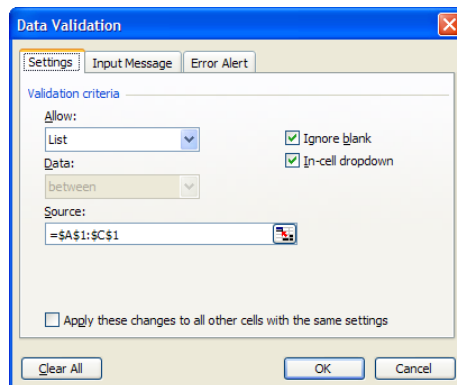


=wwsDropDown()  *with list defined in validation rule*
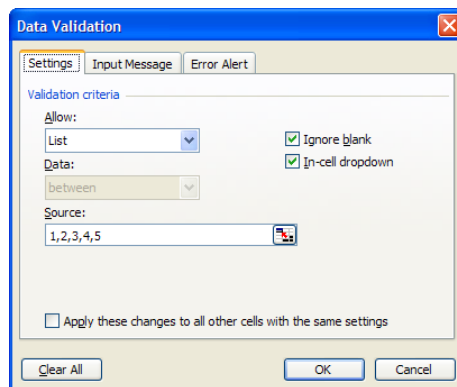


> *Tip*: Set up the validation list for the cell prior to entering the dropDown function, otherwise Excel may complain that the formula does not pass the validation rules.

**wwsCheckBox**

### Description

Creates a checkbox in the cell which the user may check or uncheck.  Multiple checkboxes may be joined in a group so that only a single option from the group may be selected.

### Arguments

*label* (required) – Text string placed next to the checkbox

*required_flag* (optional) – True or False.  If True, one of the checkboxes from a group must be selected before the webworksheet can be submitted.

*group* (optional) – checkboxes which are assigned to the same group can have only one of the checkboxes selected.  Selecting one will uncheck all the others.

*selected* (optional) – True or False.  If True, the checkbox will be checked by default.

### Validation Options

### Examples

=wwsCheckBox("Freight Included")

=wwsCheckBox("Blue", false, 1)

Assigning a group:

| Select Your Color: | =checkBox("Blue", true, 1, true) | =checkBox("Red", true, 1) | =checkBox("Green", true, 1) |
|---|---|---|---|

will produce:

| Select Your Color: | ☒ Blue | ☐ Red | ☐ Green |
|---|---|---|---|

only one of which can be selected since they are all members of group 1.

> ***Tip***: When a checkbox is selected, the value of the cell is set to the checkbox label, which can be referenced in other formulas.

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
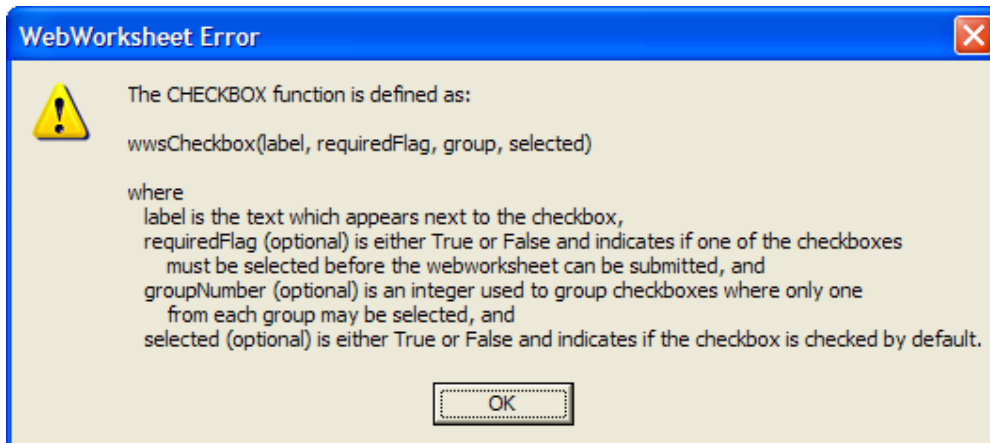
## wwsCalendar

### Description

Creates a popup calendar (date picker) for the cell. The calendar can be configured to automatically appear when the cell is active, or only when the user clicks on the calendar icon displayed next to the cell.

### Arguments

*default_date* (optional) – Text string containing a valid date which is automatically inserted into the cell.

*autoshow* (optional) – True or False. If True, the calendar icon is not shown and the date picker will appear automatically when the cell is selected (by clicking or tabbing into the cell). If False, the calendar icon is displayed and the user must click on the icon to see the date picker. If not provided, it defaults to False.

### Validation Options

User input may be validated to a specific date, or range of dates, using the standard Microsoft Excel validation functions on the Data ... Validation menu.

### Examples

=wwsCalendar()

=wwsCalendar("12/25/2011")

=wwsCalendar("7/4/2011", True)

=wwsCalendar("", True)


When the date picker is activated (either by entering the cell or clicking the calendar icon), the calendar will appear directly below the cell, and the date in the cell will be selected, or the current date if the cell has no value. Here is an example:
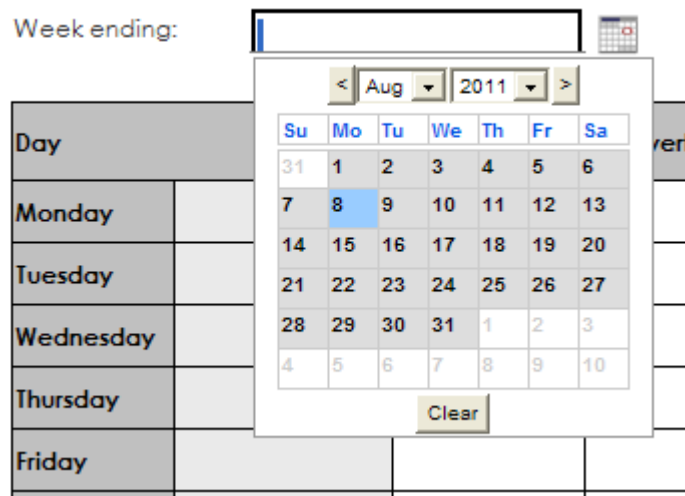


The calendar icon will be placed to the right of the input cell and will be sized to the height of the row. If text appears in the cell to the right of the input cell, the calendar icon may obscure all or part of the text.

**wwsTabOrder**

### Description

Allows the tab order to be specified instead of defaulting from right to left, top to bottom.

### Arguments

*Tab_sequence* (required) – A comma-delimited string of cell IDs which defines the sequence in which input cells will receive keyboard focus.  The sequence may also be defined as DOWN, in which case the cells receive focus in column-major order (top to bottom, left to right).

### Examples

=wwsTabOrder("B2,B4,D4,B6,B8,D7")

=wwsTabOrder("Down")

### Notes

If the row or column containing an input cell is hidden, either initially or as a result of a wwsToggle(), wwsHideRows, or wwsShowAndHide() function, focus will be given to that cell but it will remain hidden.

If the tab_sequence contains a cell which is not an input cell, it is ignored.

# Button Functions

## wwsClearButton

**Description**

Creates a button on the page which restores all the cells to their original values.

**Arguments**

*button_label* (required) – Text string which defines the text inside the button.

**Examples**

=wwsClearButton("  Clear  ")

=wwsClearButton("Start Over")

---

*Tip*: Use spaces inside the label to make a button wider to match the width of other buttons.

---

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
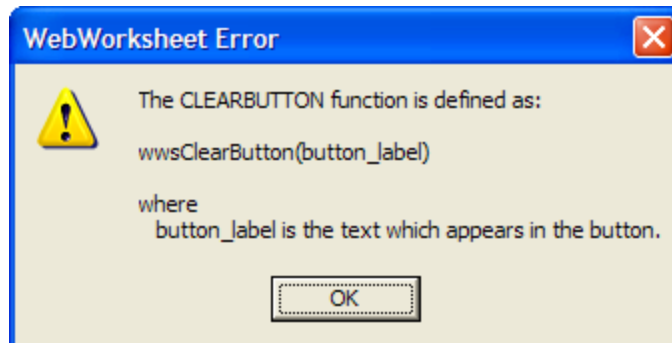
**wwsSubmitButton**

### Description

Creates a button on the page which sends the completed webworksheet to a recipient via email. The completed webworksheet contains all the values entered by the user and calculated formulas, but the mailed copy cannot be changed by the recipient. The webworksheet is contained in the body of the email message, and if the attachment name is provided, a copy is also attached.

### Arguments

*button_label* (required) – Text string which defines the text inside the button.

*email_receiver* (required) – Text string or cell reference containing the email address of the recipient of the worksheet

*email_sender* (required) – Text string or cell reference containing the email address of the sender

*email_subject* (required) - Text string or cell reference containing the subject line of the email

*attachment_name* (optional) - Text string or cell reference containing the name to be given to the attachment. WebWorksheet will name the attached file with a .htm extension.

*user_message* (optional) - Text string of message to display to the user after the form is submitted.

*next_page* (optional) - Text string which defines the next page to display after the form is submitted.

### Examples

=wwsSubmitButton("OK", "cfo@your_company.com", "its_me@your_company.com", "Weekly Timesheet", "Timesheet")

=wwsSubmitButton("OK", "cfo@your_company.com", "D5", "Weekly Timesheet", "Timesheet_" & E6, "Your timesheet has been submitted.")

=wwsSubmitButton("OK", "cfo@your_company.com;ceo@mycompany.com", "D5", "Weekly Timesheet",, "Timesheet submission complete!", "http://www.your_company.com/homepage.htm")

=wwsSubmitButton("Submit Timesheet", "ceo@mycompany.com; cc:hr@mycompany.com" & B2, "D5", "Weekly Timesheet",, "Timesheet submission complete!", "http://www.your_company.com/homepage.htm")

=wwsSubmitButton("OK", B2 & ";" & B3 & ";" & B4, "D5", "Weekly Timesheet", "Timesheet_" & E6, "Your timesheet has been submitted.")

---

*Tip*: Using the attachment_name provides the recipient with an easy method to save a copy of the completed form in a local or network folder.

---

> **Tip**: To email the completed form to multiple recipients, separate their email addresses (the email_receiver argument) with a semi-colon (;). To send the completed form as a cc: or bcc:, preface the email address with either cc: or bcc:. If using a formula to generate email lists, you must provide the semi-colon between addresses.

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:

---

**WebWorksheet Error**                                               ✕

⚠  The SUBMITBUTTON function is defined as:

wwsSubmitButton(button_label, email_receiver, email_sender, email_subject, attachment_name, user_message, next_page)

where
  button_label is the text which appears in the button,
  email_receiver is a string or cell reference containing the address where the completed form will be emailed,
  email_sender is a string or cell reference containing the email address of the person submitting the form,
  email_subject is a string or cell reference containing the text which appears in the email subject line, and
  attachment_name (optional) is a string or cell reference defining how the attached form will be named, and
  user_message (optional) is a popup message to display when the submission is complete, and
  next_page (optional) is a string defining the URL of the page to display next.

                          [ OK ]

---

**Notes**

The user_message and next_page fields can be used in conjunction to control not only what message the user sees but also where they transition to when the submission is complete. The following table describes the options:

|  | **next_page is defined** | **next_page is not defined** |
|---|---|---|
| **user_message is defined** | The user message is displayed to the user, along with a "Click here to continue" link to the next page. | Only the user message is displayed. |
| **user_message is not defined** | The user is automatically transferred to next_page as soon as the submission is complete. | The default message "Your data has been successfully submitted." is displayed. |

## wwsSaveButton

### Description

Creates a button on the page which saves the webworksheet to either a local file on the user's workstation or overwrites the original on the website with any updates. The button label will be displayed as the cell's value.

### Arguments

*button_label* (required) – Text string which defines the text inside the button.

*location* (optional) – Text string which defines the save location as either local or shared. If not provided, the default value is local.

*password* (optional) – Text string which defines the password required to save the shared file.

### Examples

```
=wwsSaveButton("   Save   ")
=wwsSaveButton("   Save   ", "local")
=wwsSaveButton("Update", "shared")
=wwsSaveButton("Update", "shared", "P@ssw3rd!")
```

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
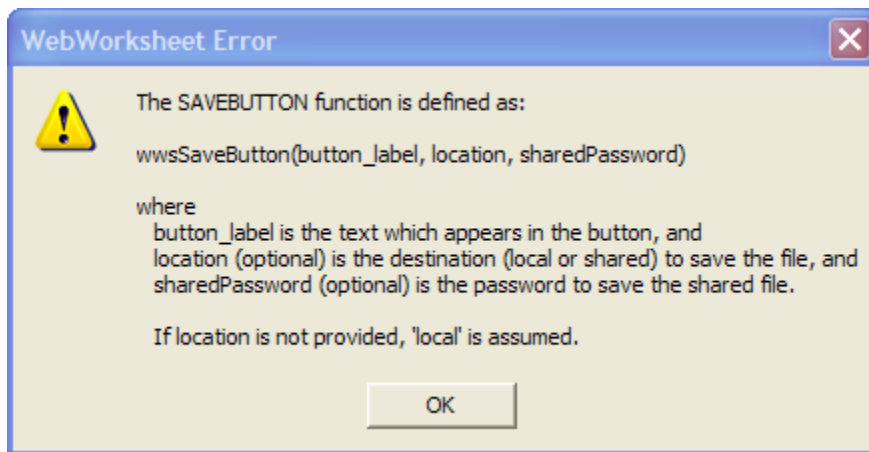


When the user clicks on the savebutton and the save_location is local, the following message will be displayed:
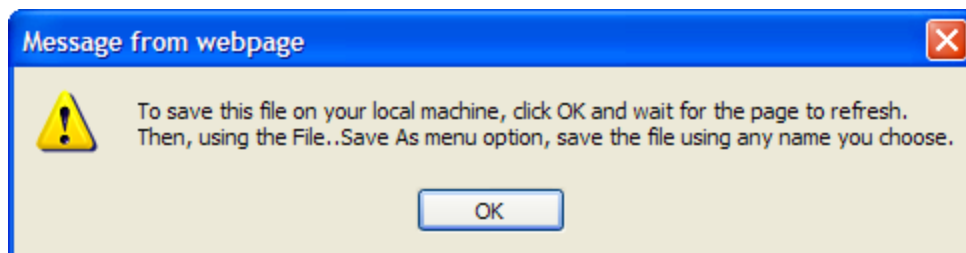


This function will work only if the user is connected to the internet, and the user must wait for the page to redisplay itself before saving the file. This may take a few seconds during periods of heavy internet traffic.

If a password is required to save the shared file, the user will be prompted to enter the password with one of the following popup screens (depending on their browser):

**Explorer User Prompt**

Script Prompt:

A password is required to save any changes:

OK

Cancel

**Javascript**

A password is required to save any changes:

OK     Cancel

If the incorrect password is entered, the following message will appear:

**Message from webpage**

⚠ Incorrect Password.  Changes will not be saved.

OK
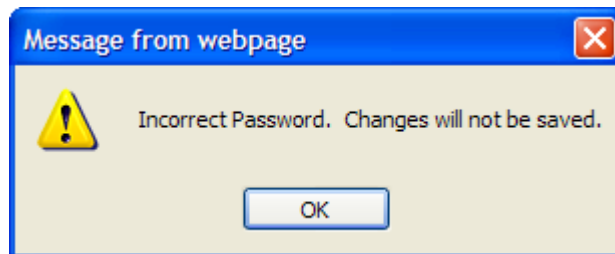
If the Cancel button is selected, the following message will appear:

**Message from webpage**

⚠ No password entered. Changes will not be saved.

OK
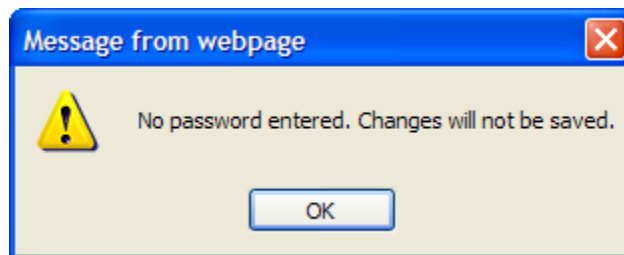
*Tip*: A shared page can be updated by any user, but if multiple users concurrently access the page, the page will reflect only the changes made by the "last" save.  In other words, the last one out wins.

**wwsPrintButton**

**Description**

Creates a button on the page which prints the webworksheet to a user-selected printer. A message can be displayed to the user to provide instructions before the Windows Print Dialogue box appears.

**Arguments**

*button_label* (required) – Text string which defines the text inside the button.

*userText* (optional) – Text message which can be displayed to the user prior to the print dialog box appearing.

**Examples**

=wwsPrintButton("Print")

=wwsPrintButton(" Print ", "Set orientation to landscape before printing.")

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:



If the userText message was defined, a popup box similar to the following will be displayed:
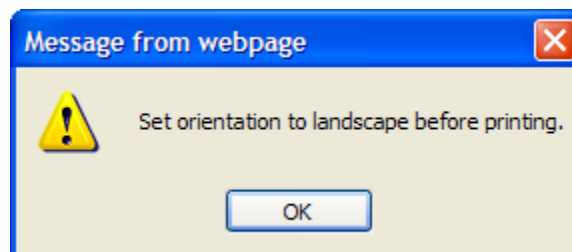
**wwsCalculateButton**

**Description**

Creates a button on the page to force calculation of some or all cells. When this button is created, automatic calculation is disabled and formulas are calculated only when this button is clicked.

**Arguments**

*button_label* (required) – Text string which defines the text inside the button.

*calculation_range* (optional) – Range of cells over which the formulas will be evaluated. If omitted, all formulas on the page are recalculated. Any formulas which are dependent upon cells in the specified range will also be recalculated.

*bookmark* (optional) – Cell reference or bookmark name to receive focus after the calculation is complete.

**Examples**

=wwsCalculateButton("Calculate")

=wwsCalculateButton("Calculate", B10:D20)

=wwsCalculateButton("Calculate Area", D5, B5)

=wwsCalculateButton("Calculate All",,namedcell)


If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
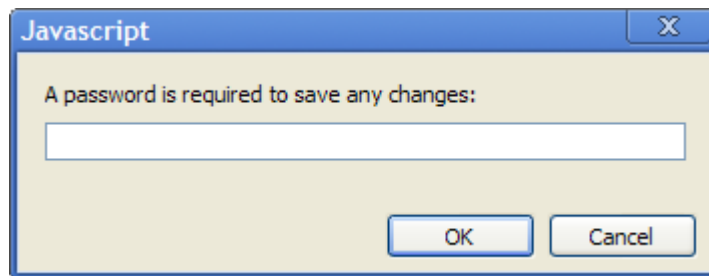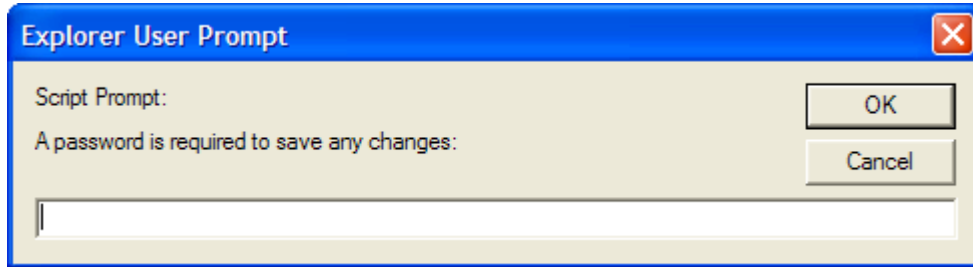


**WebWorksheet Error**

The WWSCALCULATEBUTTON function is defined as:

wwsCalculateButton(button_label, calculation_range, bookMark)

where
button_label is the text which appears in the button, and
calculation_range (optional) is a range of cells to be re-calculated, and
bookMark (optional) is the name of the bookmark or input cell to scroll into view.

[ OK ]

---
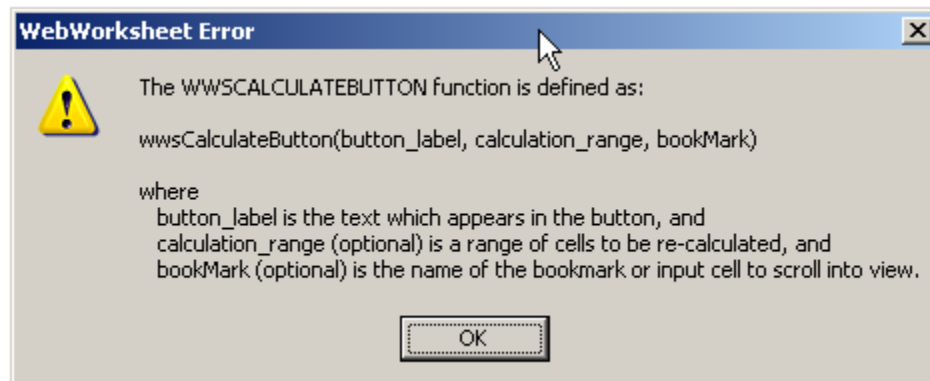
*Tip*: When a calculation_range is specified, include any input cells which provide values to the formulas in the calculation_range.

---

## wwsCodeButton

**Description**

Creates a button on the page which executes custom javascript. Use in conjunction with the "includeScript" option in the wwsSetup function to define the file containing the javascript code.

**Arguments**

*button_label* (required) – Text string which defines the text inside the button.

*function_name* (required) – Text string which defines the name and arguments passed to a custom javascript function.

**Examples**

=wwsCodeButton("Sort", "myCustomSort()")

=wwsCodeButton("   Sort   ", "sortView(3)")

---

*Tip*: In order to expand the width of this button to match others, insert spaces into the button label.

---

**Notes**

Unlike other buttons, which consume an entire row and are automatically centered, this button resides in a single cell. The cell must therefore be sized (or merged) to fit the entire button.

**wwsFileAttach**

**Description**

Creates a button on the page which allows a local file to be attached to the email.

**Arguments**

*Allowed_file_types* (optional) – Comma-delimited string of file types which may be uploaded.  If not specified, any file type may be selected for upload.

**Examples**

=wwsFileAttach()

=wwsFileAttach("xls,xlsx,doc,docx,jpg,pdf")

---

*Tip*: The wwsFileAttach function may be included more than once if multiple files may be uploaded.

---

**Notes**

If the user selects a file type which is not included in the list of allowed types, a warning message will be displayed (as shown below), but the file will still be uploaded.
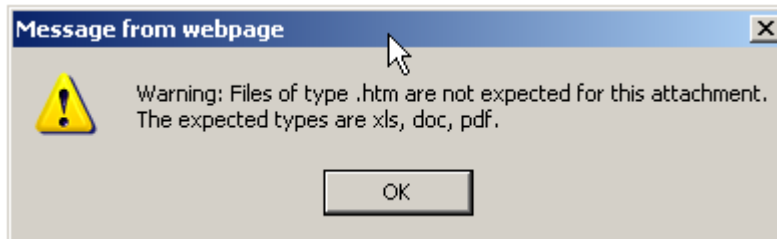
## wwsUserClicked

### Description

Returns true if the user clicked on the button named as the argument, false otherwise. This function is used to execute specific formulas only when a button is clicked, and can be used to hide rows, show rows, or set cells to defined values as a result of a button click.

### Arguments

*buttonName* (required) – Text string containing the label (name) given to the submit, save, or print button.

### Examples

To hide rows 26 to 30 when the user clicks on the button named "Submit" use:

=if(wwsUserClicked("Submit") = TRUE, wwsHideRows(ROW(A26),ROW(A30)), "")

To show rows 10 to 20 when the user clicks on any button EXCEPT the "Update" button use:

=if(wwsUserClicked("Update") = FALSE, wwsShowRows(ROW(A10),ROW(A20)), "")

To add a print date and time in cell G20 to a form before it's printed, use:

=if(wwsUserClicked("Print") = TRUE, wwsSetCell("G20", Now()), "")

---

*Tip*: This function provides the ability to change the form prior to submitting, saving, or printing the form.  For example, it can be used remove user instructions before a completed form is emailed.  If your form dynamically hides and shows rows based on user interaction, it can be used to make all the rows visible before the form is emailed.  If disjoint (non-contiguous) sections of the form need to be hidden or made visible, just use multiple formulas, one per section.

---

### Notes

This function is not available for the wwsClearButton function.  For example, if you define a Reset button on your form using:
=wwsClearButton("Reset")

and attempt to define a formula to execute when the Reset button is clicked as
=if(wwsUserClicked("Reset") = TRUE, wwsSetCell("A7", 100), "")

then cell A7 will be still be set to its original value, not to 100.

# Data Handling and Integration Functions

## wwsDbQuery

### Description

Retrieves data from a database using a SQL statement and places the result in a range of cells.

### Arguments

*SQL* (required) – Text string which defines the SQL select statement for querying the database.

*destinationRange* (required) – Text string which defines the cell or range of cells where the query results will be saved.

*noRecordsMessage* (optional) – Text string containing the message to display if no matching records were found.  If not provided, "Not found" is returned to the first cell in the destination range.

### Examples

=wwsDbQuery("select * from parts order by partnumber asc", "B10:D20")

=wwsDbQuery("select state from zipcodes where zip=" & A10, "C22", "Zip not found.")

### Notes

The connection to the database must be defined using the wwsSetup function when using this function.  See the wwsSetup() function for more details.

The number of records displayed is defined by the destination range.  If more records are returned than fit into the defined range, the last row of the range is used to provide commands for paging up and down, displaying the page number, and moving to the first and last page.  For example, if the query from the first example returned 100 records, 10 would be displayed on each page in rows 10-19, and row 20 would be used for the paging controls.

Currently, in order to use this function, your database server must allow ODBC connections to a Microsoft Access database and support either ASP or PHP server-side scripting technologies.  The interface script must reside in the same folder as the database on your web server.

The database connection information and the SQL statements are encrypted inside the HTML code to prevent users from learning about your database table or field names.

---

*Tip*: In order to ensure all the paging controls are visible, the cells of the last row in the destination range should be merged into one with center alignment.

---

*Tip*: If you are not familiar with SQL and the Select statement, the following link can provide an introduction:

http://www.w3schools.com/sql/sql_select.asp

Please be aware that Microsoft Access does not support all options available for the select statement.

---

**wwsFilter**

**Description**

Provides Excel filtering and sorting capabilities over a range of rows.

**Arguments**

*Label* (required) – Text string which defines the column header.

*dataRange* (required) – Range of cells over which the selected filter or sort will be applied.  This may be either a range of cells (e.g A2:F25) or a named range.

**Examples**

=wwsFilter("State",B4:F119)

=wwsFilter("City", A4:C20)

=wwsFilter("Part Number", partInfo)

**Notes**

The standard Excel filtering and sorting functions will be applied over the defined range. Filters for the Top 10 and Custom options are not currently supported.  Sorting is applied according to the type of format applied to the cell directly below the header. For example, given the range shown below, the Number column will be sorted as numbers if the cell below the Number header (containing 100) is formatted as a Number, or sorted as text if the cell is formatted as General.

| Number ▼ | Color ▼ | State ▼ | Dates ▼ |
|---|---|---|---|
| 100 | yellow | MI | 3/1/2014 |
| 67 | | AZ | 12/14/2013 |
| 9 | yellow | AK | 7/13/2013 |
| 3 | white | RI | 6/23/2013 |
| 4 | | 3 | 2/9/2013 |
| 4 | red | WI | 11/24/2012 |
| -12 | green | WI | 9/8/2012 |
| 3 | yellow | | 4/7/2012 |
| def | red | RI | 1/21/2012 |
| 2 | green | RI | 11/5/2011 |
| 11 | red | AZ | 9/28/2011 |
| 99 | Blue | AZ | 8/20/2011 |
| 5 | BLUE | WI | 4/27/2011 |
| 1 | blue | 77 | 3/19/2011 |
| 0 | Orange | WA | 1/1/2011 |
| | green | MI | |

If the dataRange is populated using the wwsDbQuery function, filtering and sorting is done only on the current page of data.  Paging though database records removes any filters or sorting options.

## wwsGoTo

**Description**

Moves to the web page defined by the URL, passing the data as an encrypted string.

**Arguments**

*URL* (required) – Text string which defines the name of the new HTML page, relative to the current page.  For example, to move to a page called login.htm in the same folder, URL would be set to "login.htm".  To move to a page called login.htm in a subfolder called "clients", URL would be set to "/clients/login.htm".

*data* (optional) – One or more data values to be sent to the new URL, each separated by the pipe (|) symbol.  The data string is encrypted so their actual values are not visible in the address bar.  The encrypted data is made available to the receiving page using the wwsGetUrlData() function.

**Examples**

=IF(B4="myPassword", wwsGoTo("login.htm"), "")

=IF(B4="myPassword", wwsGoTo("/clients/acme/login.htm", C10), "")

=IF(AND(userid<>"",password<>"",password=C22),wwsGoTo(VLOOKUP(userid,

userlist,3,0),D11 & "|" & A14),"")

---

*Tip*: To create a simple link to move to another page, use the Excel hyperlink command.

---

## wwsGetUrlData

**Description**

Retrieves encrypted data passed via the wwsGoTo command and places the original value in the cell.

**Arguments**

*argumentNumber* (required) – Integer defining which argument to decode and place into the cell.

**Examples**

=wwsGetUrlData(1)          *gets the first data argument*

=wwsGetUrlData(3)          *gets the third data argument*

# Image Functions

## wwsImage

### Description

Places the specified image file in the cell and sizes it the given height and width.

### Arguments

*filename* (required) – Text string which defines the file containing the image. Image types can be .gif, .png, .jpg, or .bmp.

*height* (required) – Integer defining the height of the image in pixels. If set to 0, the height of the cell will be used.

*width* (required) – Integer defining the width of the image in pixels. If set to 0, the width of the cell will be used.

*URL* (optional) – Text string which defines the address of a web page to go to when the image is clicked.

*newWindow* (optional) – Boolean used to define if URL should open in a new browser window. If missing or set to FALSE, the URL will open in the same browser tab.

### Examples

=wwsImage("companylogo.gif", 0, 0, "www.companyname.com")

=wwsImage("timesheet.jpg", 200, 300, "timesheet.htm", TRUE)

---

*Tip*: Using merged cells to define the height and width of the image will make it easier to adjust the image size to your liking instead of adjusting pixel sizes.

---

*Tip*: Use the cell comment to define text or a picture to display when the cursor is placed on the image.

---

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
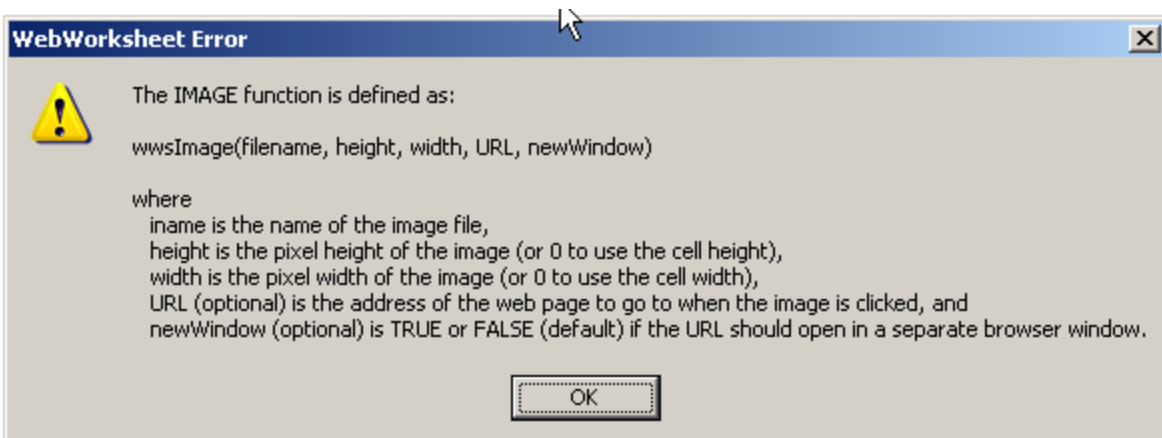
## wwsBackground

**Description**

Places the specified image file as the background image for the generated web page. The function returns the name of the image, but that name will not appear on the html page.

**Arguments**

*imageFile* (required) – Text string which defines the file containing the image. Image types can be .gif, .png, .jpg, or .bmp.

**Examples**

=wwsBackground("companylogo.gif")

=wwsBackground("http://www.mycompany.com/images/logo.gif")

---

*Tip*: The background image fills in the screen space outside the actual page and can be aesthetically pleasing or a distraction depending on the image.

---

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:

# Display Functions

## wwsToggle

### Description

Shows or hides rows when the message text is clicked.

### Arguments

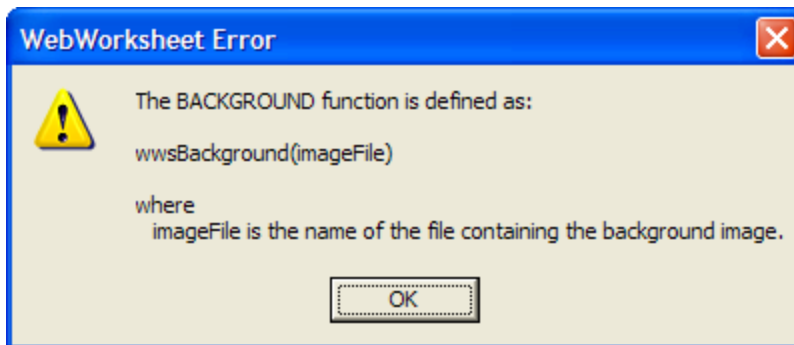*cellText* (required) – Initial text string to display in the cell. Clicking on this text causes the row(s) which follow the message to be hidden or shown.

*rowCount* (optional) – Integer defining the number of rows following the message to be hidden or shown. If rowcount is missing, only the following row will be toggled.

*cellTextWhenVisible* (optional) – Text string displayed in the cell when the toggled text is made visible. This allows the text shown on the page to change to reflect the visibility of the rows.

*cellTextWhenHidden* (optional) – Text string displayed in the cell when the toggled text is hidden.

### Examples

=wwsToggle("Click here to see a full description of this product")

=wwsToggle("Click here to show the full error message", 3)

=wwsToggle("Show Detail", 4, "Hide Detail", "Show Detail")

---

*Tip*: Defining an Excel comment for the cell containing the message will result in the comment being displayed when the mouse hovers over the message.

---

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
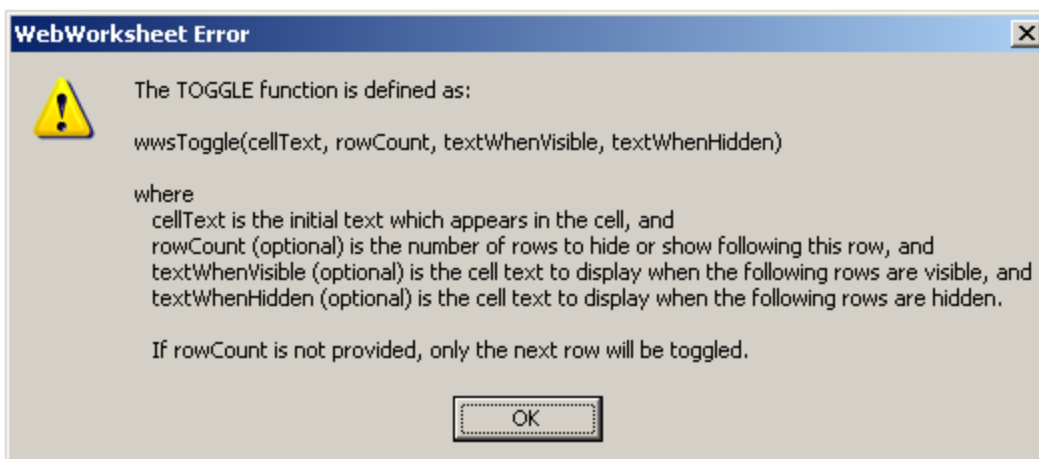
**wwsShowRows**

**wwsHideRows**

### Description

Shows or hides rows as a result of a user action or calculation.

### Arguments

*startRow* (required) – Integer which defines the first row to show or hide.

*endRow* (optional) – Integer which defines the last row to show or hide. If endRow is missing, only the startRow will be shown or hidden.

*increment* (optional) – Integer which defines which offset rows are shown or hidden (e.g. increment of 2 will show or hide every other row, 3 every $3^{rd}$ row, etc). If increment is missing, it will default to one (every row).

### Examples

=if(A59="Yes", wwsShowRows(60), wwsHideRows(60))

=if(ucase(A59)="NO", wwsHideRows(ROW(A60),ROW(A64)), wwsShowRows(ROW(A60),ROW(A64)))

=if(C60="Yes", wwsShowRows(60, 80), wwsHideRows(61, 80, 2))

> **Tip**: These functions are particularly useful when constructing intelligent forms which show and hide sections based on user input.  Hiding or showing a section can be controlled via a checkbox, dropdown list, or a calculation.  Using the ROW function as arguments will automatically adjust the row numbers as rows are added to or deleted from the Excel worksheet.

If any error is detected while using these functions in the worksheet, a message similar to the following will appear, and #ERROR! will appear in the cell:
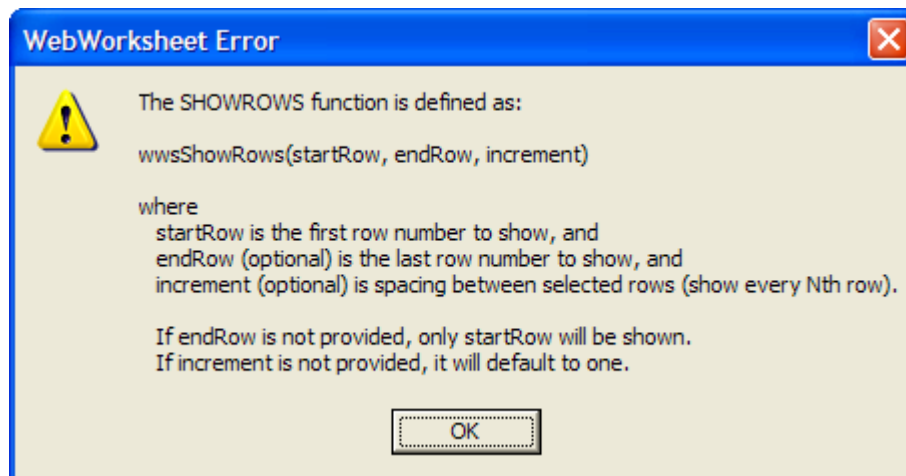
**wwsShowAndHide**

Description

Shows and hides rows as a result of a user click.

Arguments

*label* (required) – Text displayed in the cell.

*showStartRow* (required) – Integer or function which defines the first row to show.

*showEndRow* (required) – Integer or function which defines the last row to show.

*hideStartRow* (required) – Integer or function which defines the first row to hide.

*hideEndRow* (required) – Integer or function which defines the last row to hide.

*bookmark* (optional) – Cell reference or bookmark name to receive focus.

Examples

=wwsShowAndHide("More . . .", 14, 16, 13, 13)

=wwsShowAndHide("Less . . .", 13, 13, 14, 16)

=wwsShowAndHide("More . . .", ROW(A14), ROW(A16), ROW(A13), ROW(A13))

=wwsShowAndHide("Preview", ROW(A1), ROW(A20), ROW(A21), ROW(A30), A5)

=wwsShowAndHide("Less . . .", ROW(A13), ROW(A13), ROW(A14), ROW(A16), "Top")

> *Tip*: Using the ROW function as arguments will automatically adjust the row numbers as rows are added to or deleted from the Excel worksheet.

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
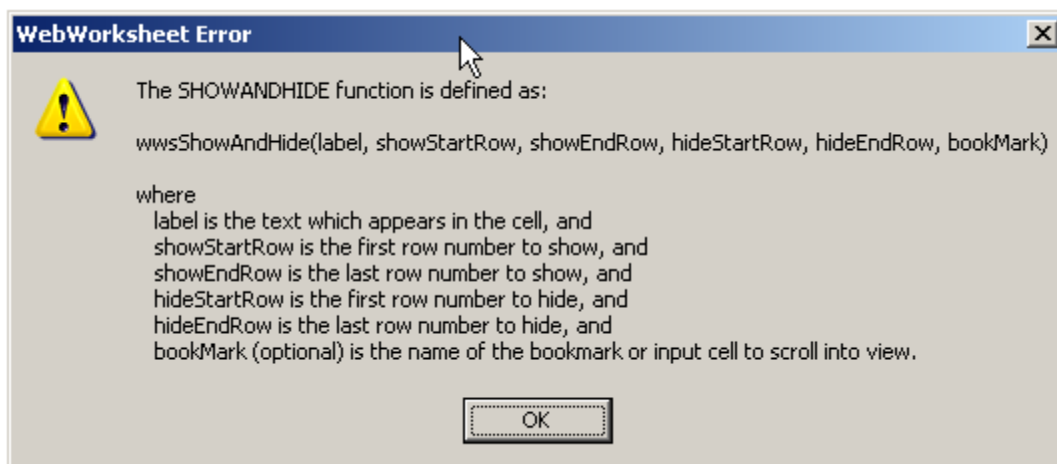


Notes

If a cell reference is given as the bookmark to scroll into view, that cell must contain an input function (e.g. wwsInput). If the row containing the cell reference or bookmark is hidden, it will remain hidden.

**wwsHide**

### Description

Provides a method for instructing WebWorksheet to hide this row in the generated file. This allows the row to remain visible in Excel to ease development, but the row will be hidden when displayed on the web.

### Arguments

*none*

### Examples

=wwsHide()

---

*Tip*: Using this function instead of manually hiding the rows will actually shorten the time it take WebWorksheet to generate the HTM file.  See Appendix A for more detail.

---

**wwsVisible**

Description

Returns true if the specific row is visible or false if hidden.

Arguments

*rowNumber* (required) – Integer or function defining the row number to check.

Examples

=if(wwsVisible(14) = true, wwsShowRows(15), wwsShowRows(16))

=if(wwsVisible(ROW(A14)) = false, wwsShowRows(ROW(A15),5), wwsHideRows(ROW(A15),5))

> *Tip*: This function can be used to show or hide discontiguous rows based on a single user action.

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
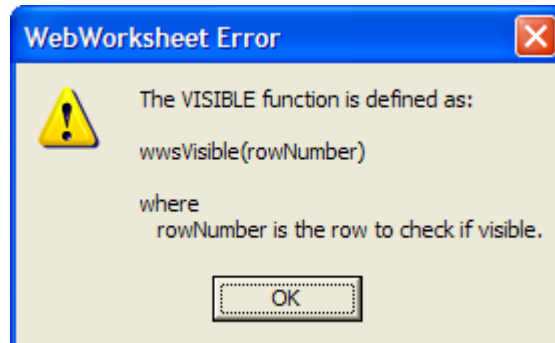
## wwsBookmark

### Description

Defines an HTML bookmark on the page which can be referenced on the same webpage or different webpages.  Bookmarks are used to automatically scroll the page to a desired location.

### Arguments

*bookmarkName* (required) – Text string containing the name of the reference bookmark.

*cellText* (optional) – Text to display in the cell.

### Examples

=wwsBookmark("Example3")

=wwsBookmark("Chapter5", "Chapter 5")

---

**Tip**: To define a bookmark when creating the hyperlink in Excel, append # followed by the bookmarkName to the address of the link.  For example, to automatically scroll the web page to Example3 when the hyperlink is clicked, define the address field of the hyperlink as:

<div align="center">MyWebPage.htm#Example3</div>

---

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:



The BOOKMARK function is defined as:

wwsBookmark(bookmarkName, cellText)

where
bookmarkName is the name given to the bookmark used as the reference, and
cellText (optional) is the text which appears in the cell.
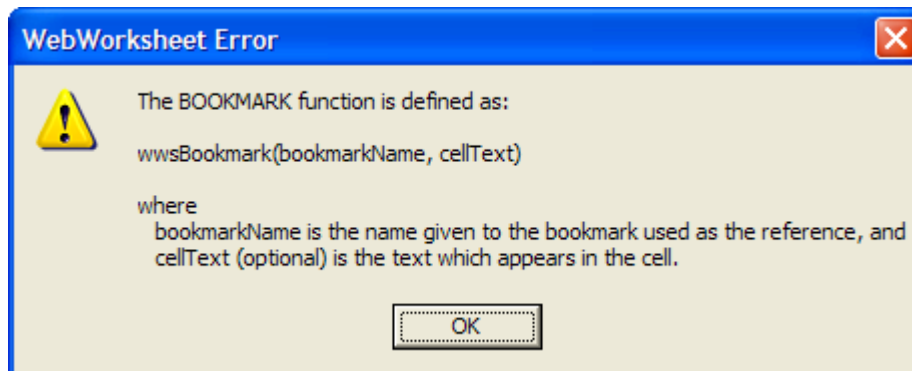
**wwsActiveBorder**

Allows the border around the active cell (the one that is currently selected) to be formatted for thickness, line style, and color.

Arguments

*borderStyle* (required) – Text string containing "*<thickness> <linestyle> <color>*"

where

*<thickness>* is the width of the border, in pixels.

*<linestyle>* is one of the valid constants used to define the style of the line, such as solid, dotted, or none.  More information on border styles can be found on the web at:

<div align="center">

[www.w3schools.com/css/css_border.asp](http://www.w3schools.com/css/css_border.asp)

</div>

*<color>* is one of the 147 valid names used to define the color of the line, such as red, blue, or green, or the hexadecimal value of a color, or the rgb function representing a color.  A list of standard color names can be found at:

<div align="center">

[www.w3schools.com/cssref/css_colornames.asp](http://www.w3schools.com/cssref/css_colornames.asp)

</div>

**Please note a space is required between each value**.

If this function is not present in your WebWorksheet, it defaults to "2px solid black" to mimic activecell highlighting in Microsoft Excel.

Examples

=wwsActiveBorder("2px solid blue")

=wwsActiveBorder("1px dotted black")

=wwsActiveBorder("2px dashed #FF0000")

=wwsActiveBorder("2px double rgb(49,106,197)")

=wwsActiveBorder("none")

> *Tip*: Use this function in conjunction with the wwsActiveBackground function to define a custom look for your WebWorksheet.

**wwsActiveBackground**

**Description**

Set the color of the interior of the active cell (the one that is currently selected).

**Arguments**

*color* (required) – Text string containing one of the 147 valid names used to define the color of the line, such as red, blue, or green, or the hexadecimal value of a color, or the rgb function representing a color.  A list of standard color names can be found at:

www.w3schools.com/cssref/css_colornames.asp

If this function is not present in your WebWorksheet, it defaults to "transparent" to allow the background color of the cell to show through.

**Examples**

=wwsActiveBackground("blue")

=wwsActiveBackground("Yellow")

=wwsActiveBackground("#FF0000")

=wwsActiveBackground("rgb(49,106,197)")

=wwsActiveBackground("transparent")

---

*Tip*: Use this function in conjunction with the wwsActiveBorder function to define a custom look for your WebWorksheet. To completely eliminate any highlighting of the active cell, set border to "none" and background to "transparent". However, any text in that cell will still be highlighted and selected when that cell becomes the active cell.

---

# Miscellaneous Functions

## wwsProtectPage

### Description

Encrypts the web page using highly secure industry-standard encryption algorithms. If a password is provided, the user must enter that same password before the page can be seen. If the password is not provided, the page is displayed immediately but the data and formulas are still encrypted.

### Arguments

*password* (optional) – Text string used to encrypt the body of the web page.

*userText* (optional) – Text string displayed on the login page. If not provided, the default prompt of "<br>This page is protected.<br><br>Please enter the password to continue: " is used.

### Examples

=wwsProtectPage()

=wwsProtectPage("Secr3tP@ssw0rd")

=wwsProtectPage("G0Packers!", "This site is intended only for the employees of Acme Packing.<br><br>Please enter the site password to continue:")

---

***Tip***: To create a "strong" password which is not easily guessed, use a combination of upper and lower case letters, numbers, and special characters.

---

***Tip***: Use the <br> tag inside the userText field to force a new line.

---

If any error is detected while using this function in the worksheet, the following message will appear, and #ERROR! will appear in the cell:
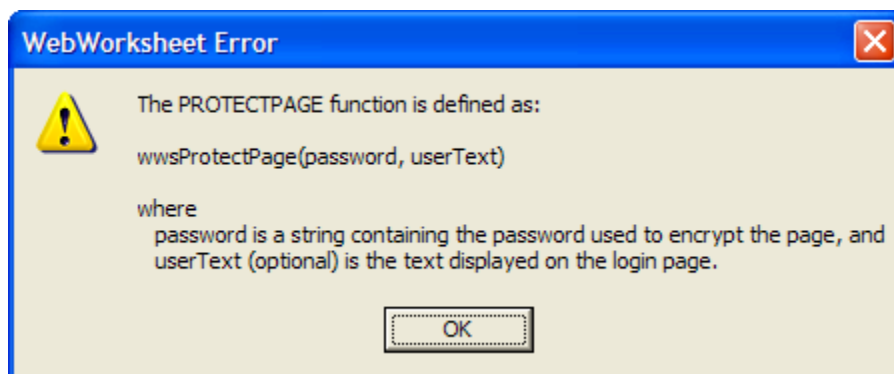
**wwsSetCell**

**Description**

Sets a specific cell to a value.  Standard Excel formulas do not allow a formula to set the value of another cell, so this provides a method for doing so.

**Arguments**

*cellID* (required) – Text string containing a valid cell identifier, such as "A12".

*value* (required) – Text string or integer value.

**Examples**

=if(wwsUserClicked("Submit") = TRUE, wwsSetCell("A13", 3), "")

=if(wwsUserClicked("Print") = TRUE, wwsSetCell("G20", Now()), "")

---

***Tip***: This function is intended to be used in conjunction with the wwsUserClicked() function to allow specific cell values to be modified as a result of a button click.

---

**Notes**

Using this function to set a cell directly (i.e. outside of its intended use), such as:

=wwsSetCell("B20", 100), or

=if(A7 > 5, wwsSetCell("B6", 100), wwsSetCell("B6", 200))

may interfere with normal formula calculations and may cause erroneous or undesirable results.

**wwsDateMath**

Description

Adds or subtracts the given number of days from the given date and returns the new date. Typically, this can be done with a simple cell formula ( e.g. =C5+3 ), but situations arise where Excel and WebWorksheet cannot agree on the cell format. Using wwsDateMath helps to resolve those situations.

Arguments

*someDate* (required) — Text string containing a valid date in mm/dd/yy or mm/dd/yyyy format.

*numberOfDays* (required) — Integer defining the number of days from *someDate*.

Examples

=wwsDateMath("12/25/2011", 3)

=wwsDateMath("1/1/2000", -180)

***Tip***: Use this function only if #VALUE! appears in a webworksheet cell which contains a date calculation formula.

**wwsDateDiff**

Returns the number of days between two dates. Typically, this can be done with a simple cell formula ( e.g. =C5–C4 ), but situations arise where Excel and WebWorksheet cannot agree on the cell format.  Using wwsDateDiff helps to resolve those situations.

Arguments

*firstDate* (required) – Cell reference containing a valid date.

*secondDate* (required) – Cell reference containing a valid date.

Examples

=wwsDateDiff(A12, B12)

=wwsDateDiff(B12, A12)

> *Tip*: If the secondDate is before the firstDate, a negative number is returned. If both dates are the same, zero is returned.

**wwsSetup**

### Description

Provides a method for changing the default settings for WebWorksheet.

### Available Settings

*iterations* – Integer defining the number of iterations over all of the formulas.  If not specified, the formulas will be evaluated from top to bottom 3 times.  Workbooks with complex formulas which reference cells containing other formulas may require a higher number of iterations.  Only increase the iterations above 3 if the WebWorksheet is not calculating correctly.  The higher the number, the longer it will take for the page to load and update after each cell change.

*includeScript* – String defining the name and location of a text file containing custom javascript to include in the generated HTML file.  This could be used to allow references to custom functions, such as those created from converted VBA macros.

*database* – String defining the name, userid, and password to a database file which can be queried using the wwsQueryDB function.  This information is encrypted to prevent users from seeing this information.

*DbQueryScript* – String defining the name of the server-side script used to process the database query requests.  This is used only when the default PHP script provided with WebWorksheet is not supported on your web server.

*calculateOnInit* – Boolean (true or false) which allows automatic calculation of all formulas to be disabled for the initial page load.  Defaults to TRUE if not specified.  Typically used with the manual calculation option.

### Examples

=wwsSetup("iterations", 1)

=wwsSetup("iterations", 5)

=wwsSetup("includeScript", "myFunctions.js")

=wwsSetup("includeScript", "http://www.mycompany.com/scripts/calculator.js")

=wwsSetup("database", "zipcodes.mdb,,")

=wwsSetup("database", "inventory.mdb,system,Adm1n")

=wwsSetup("DbQueryScript", "webworksheetDbQuery.asp")

=wwsSetup("calculateOnInit", false)

> **Tip**: If your WebWorksheet is used primarily for data entry (e.g. an order form), setting the iterations to 1 will result in faster page loads.

# Technical Notes

This section provides additional tips and insights on converting your Excel workbooks to WebWorksheet pages, and any existing limitations on Excel formulas.

**Numeric Precision**
WebWorksheet has been designed to mimic the precision of calculations as displayed in Excel. Calculation cells which are formatted in Excel as General will display up to 9 digits of precision (after the decimal point), although they are stored internally with greater precision. WebWorksheet will round numeric calculations to 9 places and display according to the format defined for the cell (9 for General, or the defined number of places if Number).

**Operator Precedence**
Excel has a defined hierarchy for evaluating formulas, which can be reviewed here. WebWorksheet utilizes this same hierarchy, but to ensure proper evaluation of complex formulas, we suggest adding parenthesis in the Excel formula to clearly identify the intended relationships and computations.

**String Comparison**
Excel is case insensitive (case does not matter) when comparing strings, but javascript, which is generated by WebWorksheet, is case sensitive. Formulas such as =IF("abc" = "ABC", "True", "False") will return TRUE in Excel and false in javascript. When creating formulas, be cognizant of case, or convert text to all one case, such as =IF(UPPER("abc") = UPPER("ABC"), "True", "False").

**Lookup Functions**
The Match, Index, Hlookup, and Vlookup functions do not support the Array form, where the lookup_array is defined in the function call ( e.g. =MATCH("b",{"a","b","c"},0) ).  The lookup_array must be defined as a range ( e.g. MATCH("b",A1:B3,0) ).  Wildcards are not supported in the Match function.

**Substitute Function**
The Substitute function will replace the first or all occurrences only.  If you specify a specific occurrence other than 1, the #FUNCTION error will be returned.

**Cells Formatted as Time**
Time-related functions, such as NOW() or TODAY(), will generate and display the current local time for the end user, adjusted for their time zone.  Hard-coded values, such as 1:45 PM, are not adjusted to local time when displayed.

**IF Statements Not Allowed as Arguments to Functions**
Excel allows IF statements to be used as argument to functions.  For example:
=VLOOKUP("Rate",H3:J6,IF(B1=2011,1,2),FALSE)
This is not supported by WebWorksheet, and the formula must be broken apart to put the IF statement as a formula in its own cell.

**Known Issues**
To find a detailed list of known issues with WebWorksheet and workarounds, look on the Support page of the website at http://www.webworksheet.com/webworksheet_support.htm. This page is accessible only to WebWorksheet customers and requires entry of your license key.

# Appendix A.  Performance Improvement Tips

While WebWorksheet has been designed to generate very efficient web pages which are compliant with current industry standards, there are some things that will improve the HTM generation time and the page loading and execution times.

1. **Reduce the number of rows and columns**
   Since WebWorksheet must read the properties (font, size, colors, etc) of every cell inside the #end marker, having fewer rows and columns will reduce processing time.  Adjust the row heights and column widths to eliminate the empty rows and columns used solely for spacing.  Merging cells together is a great way to reduce the overall number of cells.  This will also reduce the size of the generated HTM file, and therefore will reduce page load times.

2. **Place lists used solely for dropdown values outside the #end marker**
   Since the lists of values used for dropdowns are needed only when the page is created, they may be placed outside the #end marker, and will not create unnecessary rows in the HTM page.  Dropdown lists which are dynamically created must reside within the #end marker (see Appendix C).

3. **Set iteration=1 using wwsSetup if your worksheet is used for form input only**
   If your worksheet has no formulas, or just a few simple ones, setting the iteration count to one using =wwsSetup("iterations",1) will result in faster page loads and submit times.

4. **Limit the amount of font changes in a single cell**
   Excel will allow you to change the font family, color, size, and decoration of a subset of characters within a single cell.  When possible, put different formats into separate cells and set the format at the cell level instead of the character level.

5. **Use the wwsHide() function to identify rows to be hidden**
   For some unknown reason, Excel takes substantially longer to extract the properties of a hidden cell than a visible one.  Keeping all rows visible will also make it easier to develop and debug your worksheet.  Just put the wwsHide() function in one of the cells in each row you want hidden on the generated web page.

6. **Use hidden rows to contain your data or calculations instead of hidden columns**
   Browsers seem to have a difficult time with hidden columns more so than hidden rows.  For example, it's very simple to make a row visible or hidden on the web, but there is no analogous function for columns.  And browsers sometimes get confused about row heights when adjoining columns are hidden.

# Appendix B.  Validating Checkboxes before Emailing a Form

Since Excel does not natively support checkboxes as input cells, like it does for dropdown lists, the validation process to ensure one or more was checked prior to sending an email is a little different. This describes how to add validation checks for cells containing the wwsCheckbox function.

If you have a single checkbox that must be checked before the form can be submitted, set up a validation rule that simply checks the value of the cell has a length greater than one.  For example, if you use the function:

> =wwsCheckbox("I accept the terms and conditions", TRUE, 0,TRUE)

set up the validation rule as:
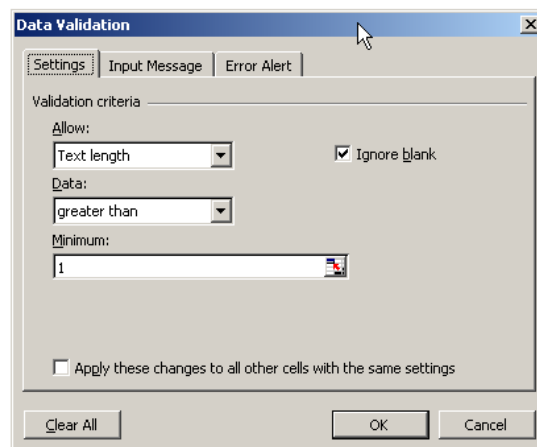


If you have a series of checkboxes, and one option has to be selected before the form can be submitted, then a validation rule is set on just one of those cells.  For example, if you define the functions in cells C14, C15, and C16 as:

> =wwsCheckbox("Agree", TRUE, 1)
> =wwsCheckbox("Neither Agree or Disagree", TRUE, 1)
> =wwsCheckbox("Disagree", TRUE, 1)

set up a validation rule only for the first cell (C14) with the custom formula:
> =OR(C14<>"",C15<>"",C16<>"")
which verifies one of the options must be checked before the submitting the form.
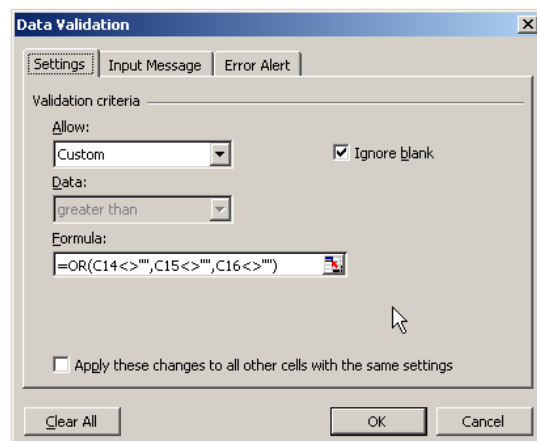
# Appendix C.  Creating Dependent Dropdown Lists

It is possible to create dependent dropdowns using WebWorksheet, where the option values displayed in one dropdown can change based on the selection in another.  The controlling dropdown is referred to as the parent, and the dependent as the child.

The data for both dropdowns is contained in a rectangular group of cells, which must be defined within the #end marker since the values change dynamically.  The rows containing this data may be hidden if desired.

Typically, the values for the parent are contained in a single column.  For each parent option, another column is defined which contains the dependent options.  The parent option must be the first value in each column for the child.  Here's an example:

| | A | B | C | D |
|---|---|---|---|---|
| 1 | | **Writing** | **Science** | **Math** |
| 2 | **Writing** | Select | Select | Select |
| 3 | **Science** | Creative Writing | Physical Science | Algebra |
| 4 | **Math** | Narrative Writing | Chemistry | Geometry |
| 5 | | Poetry | Biology | Calculus |
| 6 | | | Earth Science | |
| 7 | | | | |
| 8 | Parent: | Writing ▼ | | |
| 9 | Child: | Select ▼ | | |

The parent options are those in Column A, with the dependents in Columns B-D.  When the user selects Science in the parent dropdown, the child options are changed to those in Column C.  The formula placed in the parent cell B8 is:

    =wwsDropDown(A1:D6, B9)

where B9 is the cell containing the child dropdown, and the formula for the child dropdown is:

    =wwsDropDown()

Validation rules must also be defined for each of the dropdown cells.  The validation rule for B8 is a list with a source of A2:A4, and B9 is a list with a source of B2:B5 or whatever the defaults are.