WebWorksheet ™

Version 3.7

User Manual

Contents

Installing WebWorksheet

WebWorksheet is distributed as a standard Add-In for Excel, much like the Analysis Toolpak or Solver Add-In that are typically installed by default. However, the WebWorksheet add-in must be installed on your computer.

Follow these steps to install the WebWorksheet add-in:

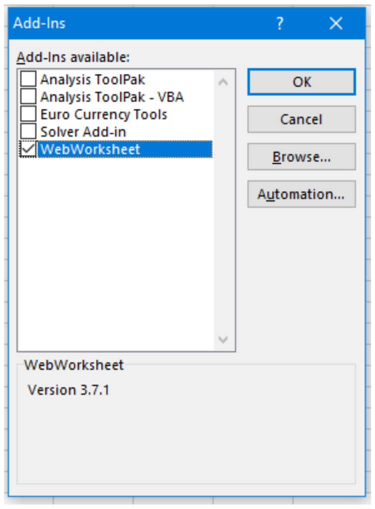
1. Download and save the self-extracting executable using the instructions provided in the confirmation email you received after the purchase was completed. This file can be saved anywhere on your computer.

2. Extract the WebWorksheet files by double-clicking on the downloaded file. Files will be installed in your %APPDATA% folder. The actual folder name where WebWorksheet is installed will be shown when installation is complete, and can always be found in the About option in the WebWorksheet menu.

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

   In Microsoft Excel 2007 and later, start Excel, then navigate to File…Options. Select the Add-Ins tab, 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 in the folder where Excel expects to find AddIns. Open the WebWorksheet folder and select webworksheet.xla, and click Open.

4. 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 Webworksheet folder and click OK.

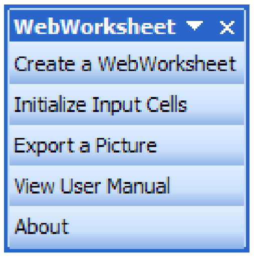


- [ ] Analysis ToolPak
- [ ] Analysis ToolPak - VBA
- [ ] Euro Currency Tools
- [ ] Solver Add-in
- [✓] WebWorksheet

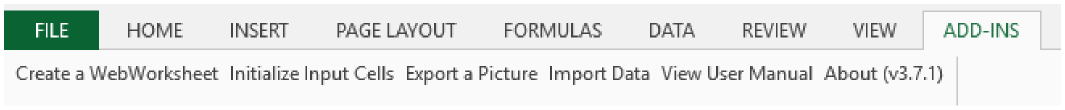
WebWorksheet
Version 3.7.1

5. When installation is complete, the WebWorksheet toolbar will now be visible in the Add-Ins tab. 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.

When installed, the following custom toolbar will appear in the Add-Ins tab:



Office 2003



Office 2007 and later

Menu Options:

- **Create a WebWorksheet** – this is the command to convert this workbook to its equivalent HTML and javascript file.
- **Initialize Input Cells** – adds the wwsInput function to all selected cells. If a selected cell has a value, it will become the default value for the wwsInput function.
- **Export a Picture** – exports pictures contained in each selected cell to a standalone .png file.
- **Import Data** – If a HTML page was created that stored cell values in a shared file, those values can be imported back into the Excel worksheet. Please note that importing values back into a worksheet overwrites any values or formulas.
- **View User Manual** – displays the WebWorksheet user manual (this document).
- **About** – displays information about the installed version of WebWorksheet.

The installation folder contains not only the addin software (webworksheet.xla), but the user manual, a folder containing samples of worksheets that have been converted into HTML files, and other related files that may be needed as described later in this manual.

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.

Please Note: WebWorksheet provides custom functions for web-enabling your worksheet, many of which have required and/or optional arguments. Since the arguments are defined by their position, you must use "" or ,, to hold the place of optional arguments if following arguments are present. For example, suppose the following function exists:

```
=wwsCustomFunction(required1, optional2, optional3)
```

The following calls are valid:

```
=wwsCustomFunction(required1)
=wwsCustomFunction(required1, optional2)
=wwsCustomFunction(required1, optional2, optional3)
=wwsCustomFunction(required1,"",optional3)
=wwsCustomFunction(required1,,optional3)
```

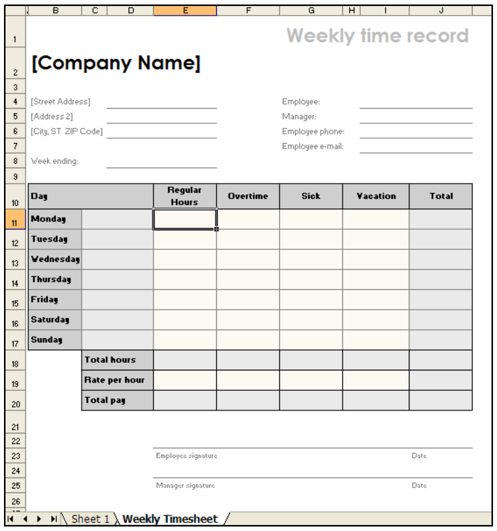
while the following calls are invalid:

=wwsCustomFunction(required1,optional3)	*missing placeholder for optional2*
=wwsCustomFunction("", optional2)	*required argument not defined*

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 that 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.



Weekly time record

[Company Name]

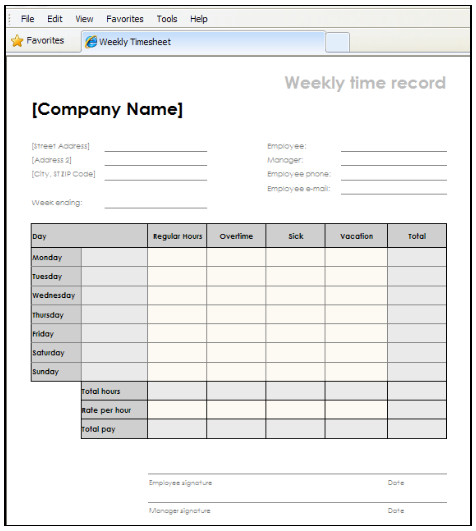
[Street Address]
[Address 2]
[City, ST ZIP Code]
Week ending:

Employee:
Manager:
Employee phone:
Employee e-mail:

Day		Regular Hours	Overtime	Sick	Vacation	Total
Monday						
Tuesday						
Wednesday						
Thursday						
Friday						
Saturday						
Sunday						
	Total hours					
	Rate per hour					
	Total pay					

Employee signature Date
Manager signature Date

Worksheet Tab Names



Weekly time record

[Company Name]

[Street Address]
[Address 2]
[City, ST ZIP Code]
Week ending:

Employee:
Manager:
Employee phone:
Employee e-mail:

Day		Regular Hours	Overtime	Sick	Vacation	Total
Monday						
Tuesday						
Wednesday						
Thursday						
Friday						
Saturday						
Sunday						
	Total hours					
	Rate per hour					
	Total pay					

Employee signature Date
Manager signature Date

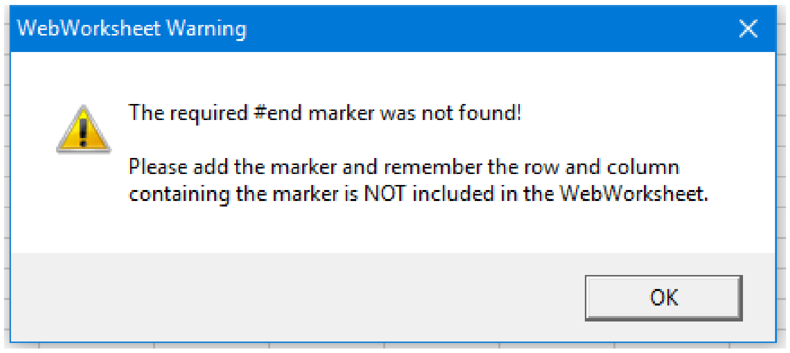
Browser Tab Names

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 that contain the marker are **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 required #end marker was not found!

Please add the marker and remember the row and column containing the marker is NOT included in the WebWorksheet.

The maximum size of the WebWorksheet is 2000 rows and 256 columns (A1:IV2000).

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 Excel workbook, 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.

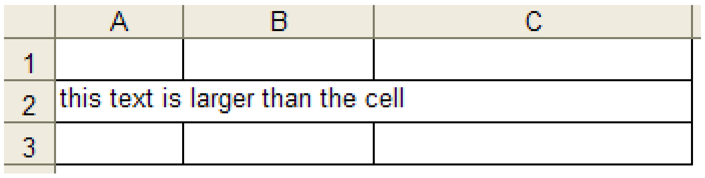
Borders: Verify the cell borders are the correct color, thickness, and style (e.g. solid or dashed). Oftentimes in a worksheet we rely on the gridlines to provide the visible borders, but gridlines are not displayed on the web version. Use the Font Group or Format Cells menu to change the borders for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

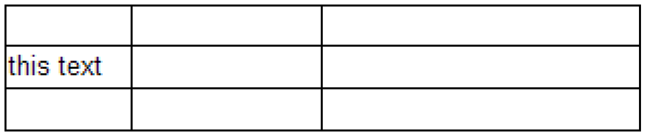
Merging Cells: If any text appears cut-off or missing on the webworksheet, it's probably because it does not fit in the cell with its current settings for font style or size. Again, Excel is forgiving in this regard, and will show the text if the adjacent cell is empty. The web cannot do that. The easiest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



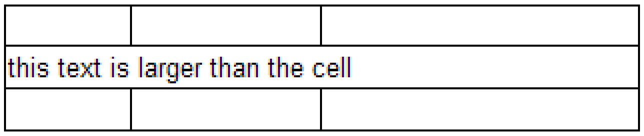
	A	B	C
1			
2	this text is larger than the cell		
3			

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



this text		

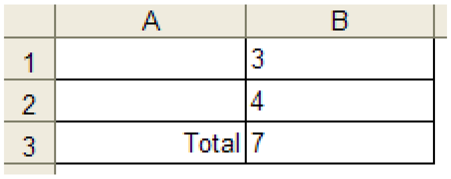
Merging cells A2, B2, and C2, and then recreating the webworksheet, will yield:



this text is larger than the cell		

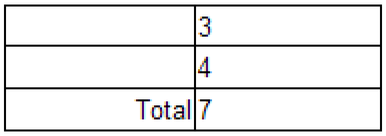
Cell Alignment: Excel oftentimes makes assumptions on whether the text in a cell should be left-aligned or right-aligned depending on the type of data in the cell (e.g. a date, a number, a text string). To create a professional looking web page, you may want to explicitly set the alignment by using the Alignment Group or Format Cells menu. 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:



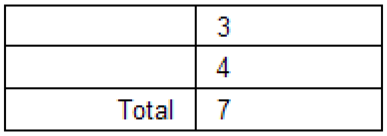
	A	B
1		3
2		4
3	Total	7

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.

Note: If the cell horizontal alignment is set to Center Across Selection, WebWorksheet automatically merges those cells and sets the alignment to Center.

Zoom Level: Excel allows you to zoom in and out on the worksheet, and this zoom level is also used by worksheet to scale the size of the generated HTML page. For example, if the zoom level is set to 50% when the Create a WebWorksheet option is selected, the resulting page will be scaled by 50%. This is useful when embedding your webworksheet into an existing web page. The scale of the generated page can be set independent of the Excel zoom using the wwsSetup() function.

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.

Tip: To use symbolic characters inside cells, select them from the Lucida Sans Unicode font as that font is widely available on computers.

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 methods which best meet your needs.

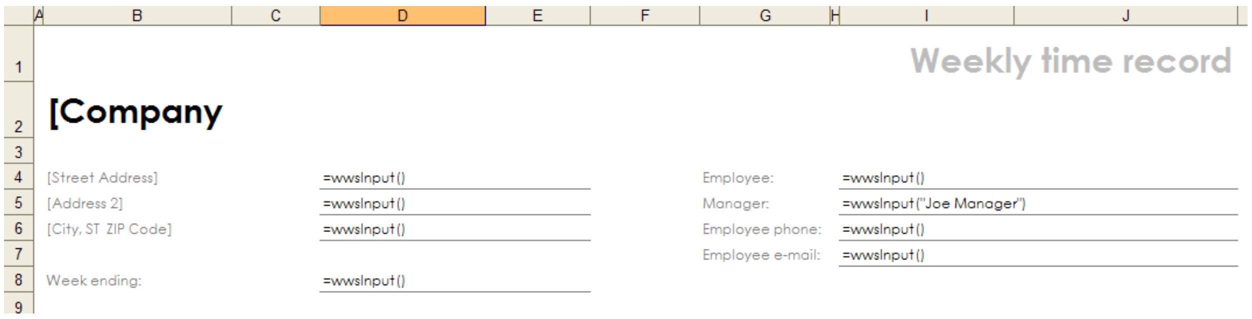
- Use the wwsInput() function to collect information to be typed by the user, including numbers, dates, and text strings, or

- 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:

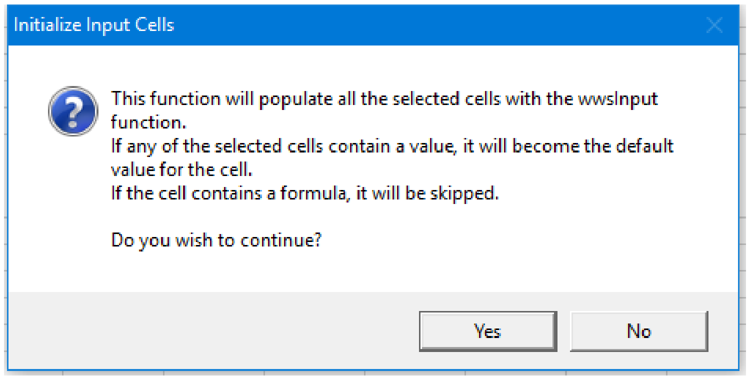


	A	B	C	D	E	F	G	H	I	J
1									Weekly time record	
2		[Company								
3										
4		[Street Address]		=wwsInput()			Employee:		=wwsInput()	
5		[Address 2]		=wwsInput()			Manager:		=wwsInput("Joe Manager")	
6		[City, ST ZIP Code]		=wwsInput()			Employee phone:		=wwsInput()	
7							Employee e-mail:		=wwsInput()	
8		Week ending:		=wwsInput()						
9										

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:

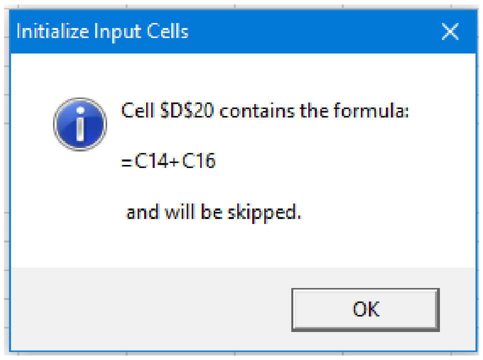


Initialize Input Cells

This function will populate all the selected cells with the wwsInput function.
If any of the selected cells contain a value, it will become the default value for the cell.
If the cell contains a formula, it will be skipped.

Do you wish to continue?

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:


Initialize Input Cells

Cell $D$20 contains the formula:

=C14+C16

and will be skipped.

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

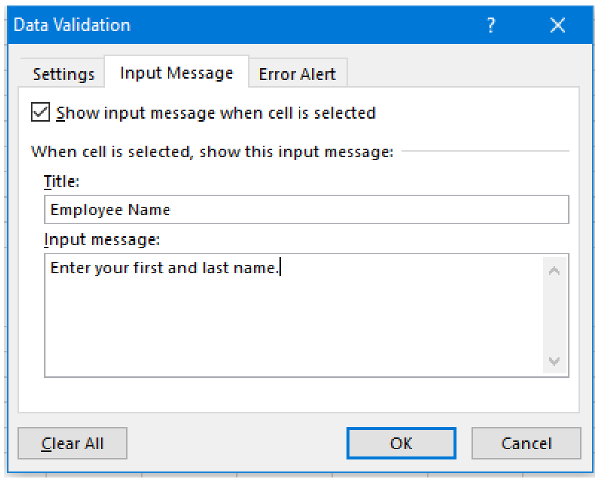
Tip: When inserting the WebWorksheet custom functions into a cell, you can use the Insert Function command in Excel to be prompted for each argument. You will find the WebWorksheet functions under the User Defined category. Also, if you enter the function name without arguments, such as =wwsCheckbox(), a dialog box will be displayed that summarizes the function and its arguments.

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:


Data Validation

☑ Show input message when cell is selected

When cell is selected, show this input message:

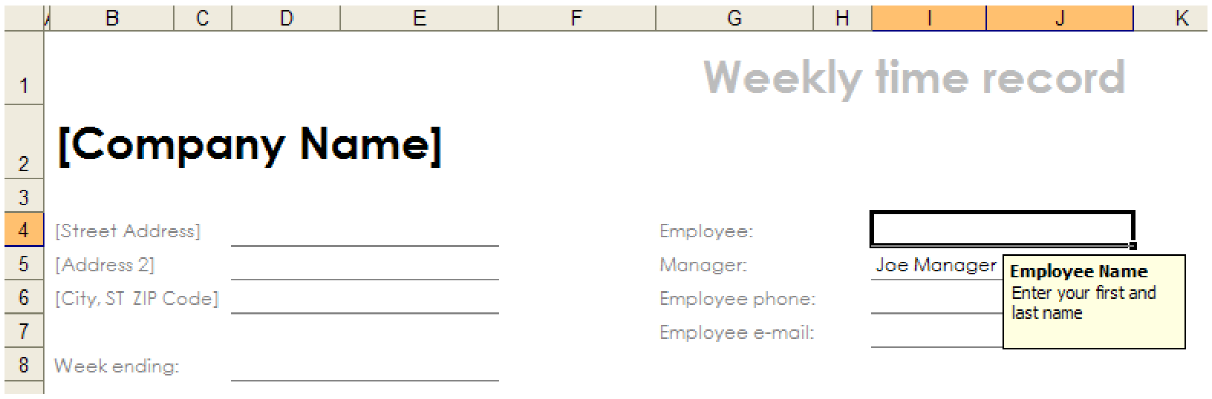
Title:

Employee Name

Input message:

Enter your first and last name.

When the Input Message is defined, Excel will show the prompt whenever that cell is active.



Weekly time record

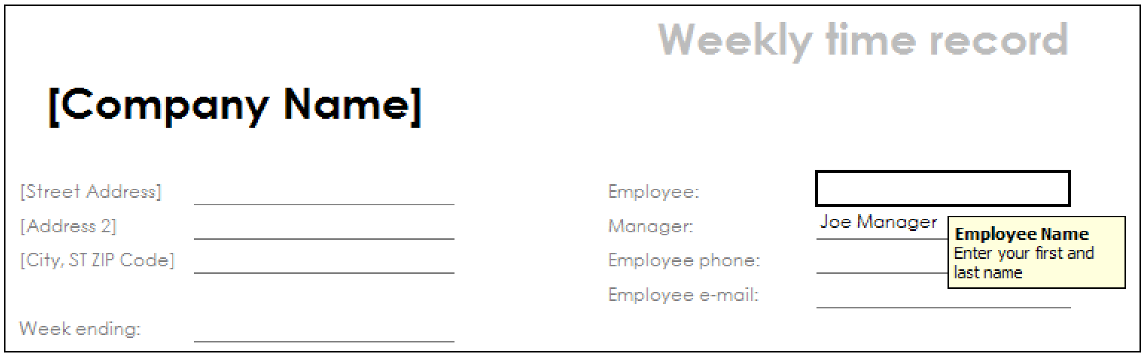
[Company Name]

[Street Address]
[Address 2]
[City, ST ZIP Code]
Week ending:

Employee:
Manager: Joe Manager
Employee phone:
Employee e-mail:

Employee Name
Enter your first and last name

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



Weekly time record

[Company Name]

[Street Address]
[Address 2]
[City, ST ZIP Code]
Week ending:

Employee:
Manager: Joe Manager
Employee phone:
Employee e-mail:

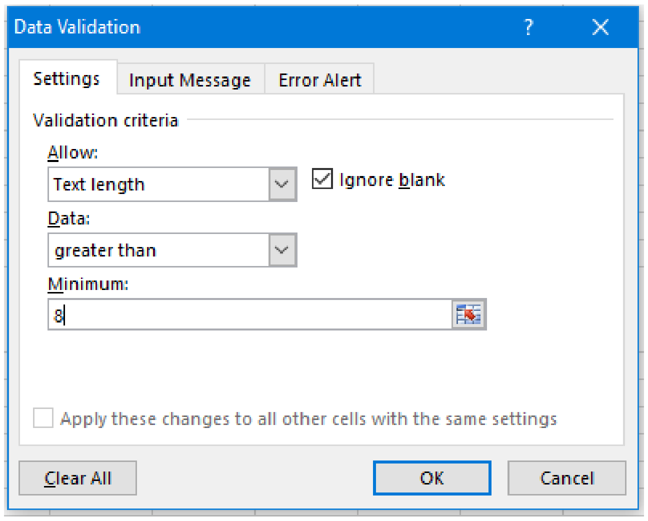
Employee Name
Enter your first and last name

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 on the above timesheet form. The Settings tab would be used to define a minimum length for the name, such as:



Validation criteria

Allow: Text length

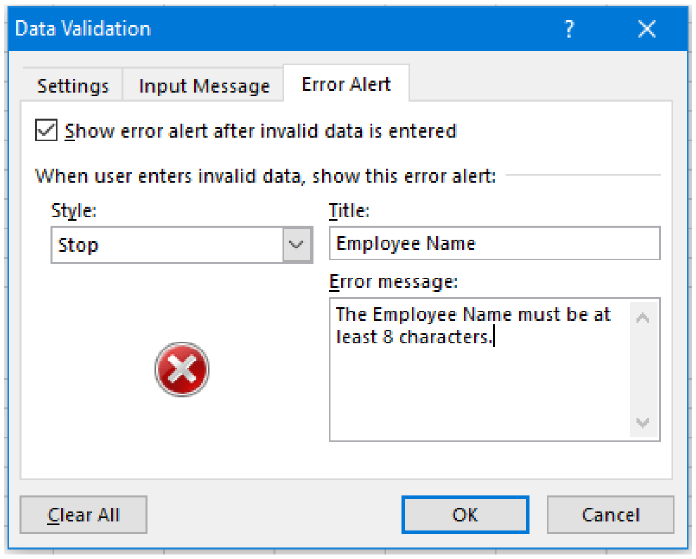
Ignore blank [✓]

Data: greater than

Minimum: 8

Apply these changes to all other cells with the same settings [ ]

Then the Error Alert tab would be set up as:



Show error alert after invalid data is entered [✓]

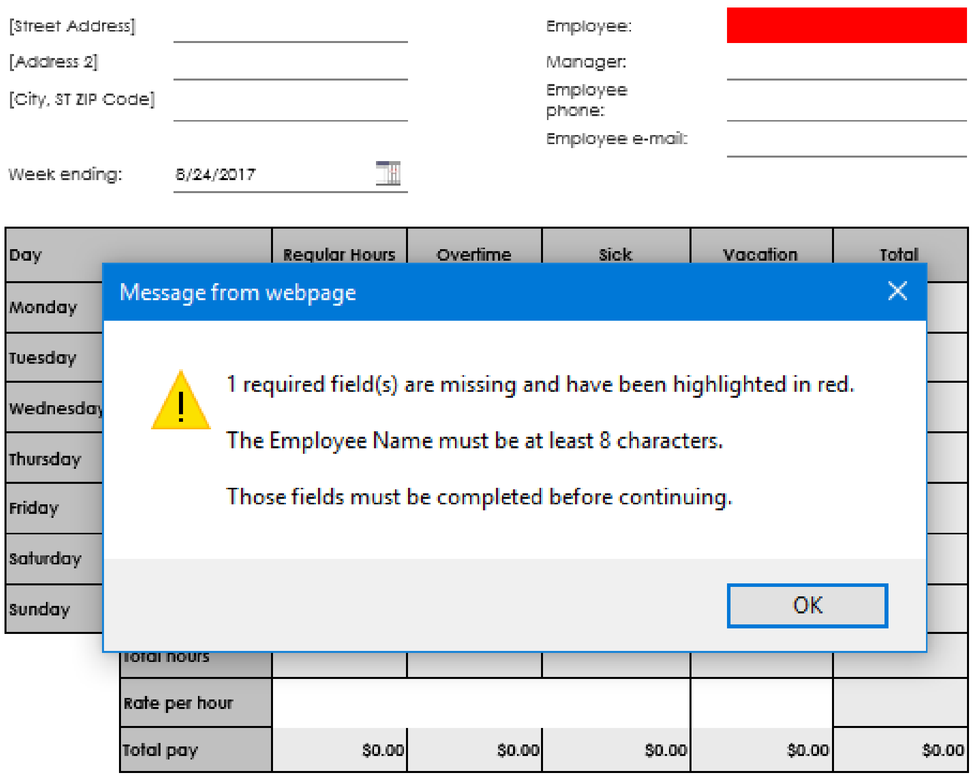
When user enters invalid data, show this error alert:

Style: Stop

Title: Employee Name

Error message: The Employee Name must be at least 8 characters.

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

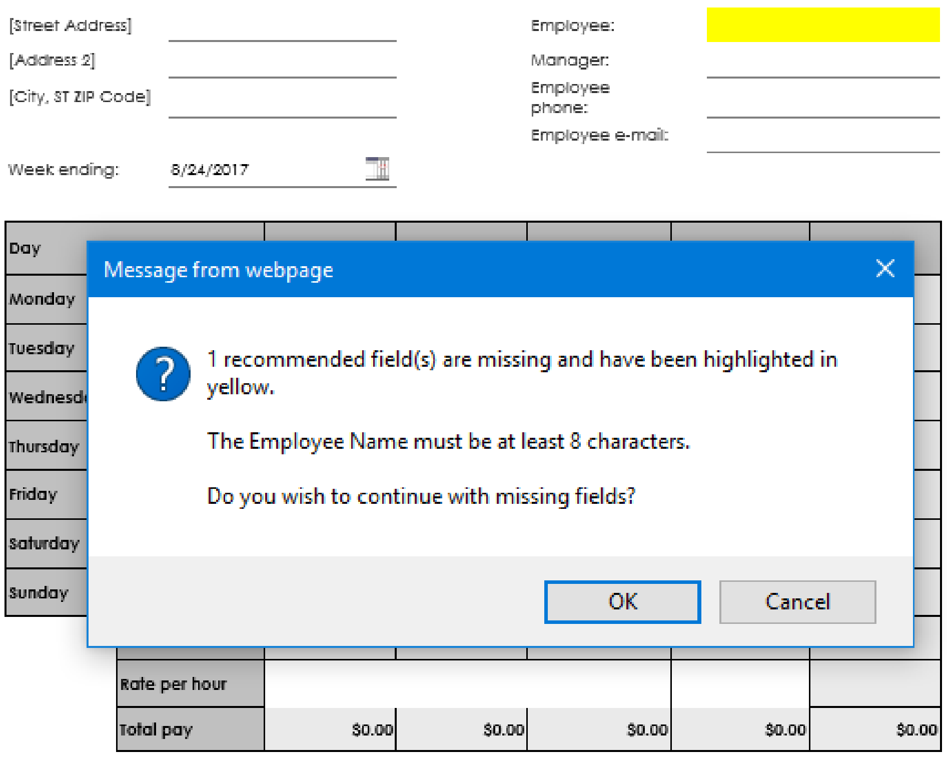


1 required field(s) are missing and have been highlighted in red.

The Employee Name must be at least 8 characters.

Those fields must be completed before continuing.

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



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

The Employee Name must be at least 8 characters.

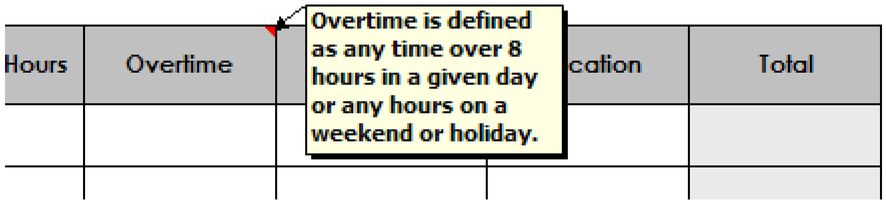
Do you wish to continue with missing fields?

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 page is submitted as is.

Defining Mouseover Effects

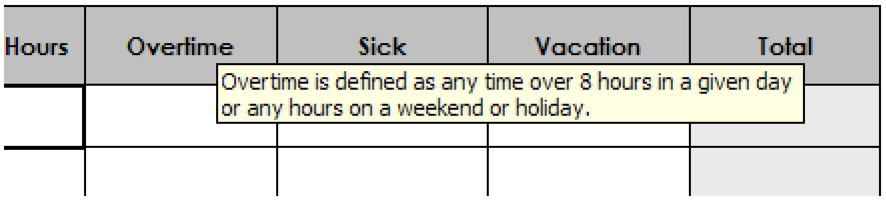
A mouseover can be defined for any cell on the webworksheet by defining a comment for that cell in Excel. Using the Insert Comment menu option, define the message to be displayed on the webworksheet whenever the user hovers the mouse over that field. The red indicator that Excel uses to identify cells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

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



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:



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

Tip: To display a mouseover for a button, define a comment for the cell containing that button function.

Highlighting the Active Cell

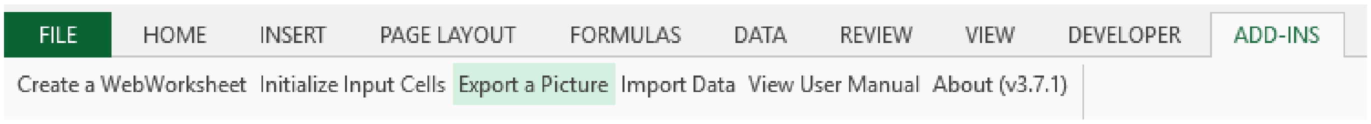
By default, Microsoft Excel places a thick, black border around the active cell (the cell currently selected). While this works well within Excel, it may not be the ideal or desired way to identify the active input cell on the web. Therefore, WebWorksheet provides functions for you to customize the active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

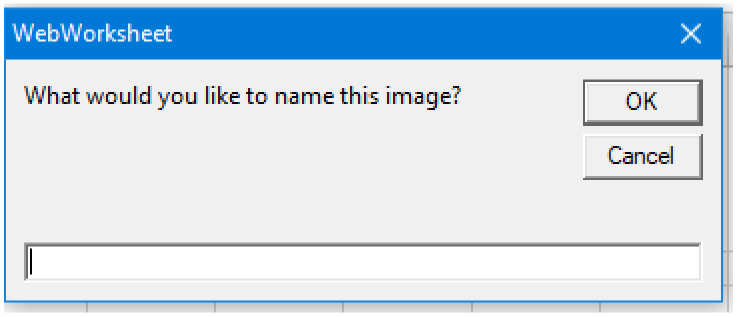
Including Images in your WebWorksheet

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. If the image already exists as a standalone image file (e.g. a .gif or .jpg file), it is not necessary to extract the image from the worksheet.

If the image exists only within the worksheet, it must be manually extracted to create a standalone image file. 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.

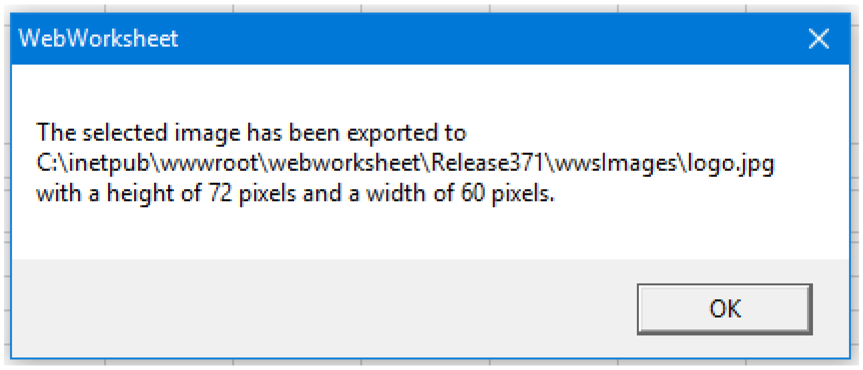


When an image is selected, you will be prompted to enter a filename to store the extracted image:


WebWorksheet

What would you like to name this image?

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.

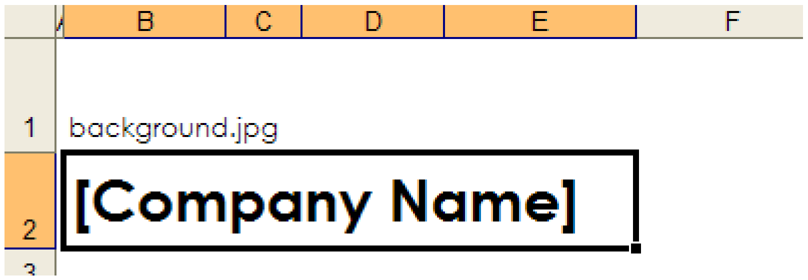

WebWorksheet

The selected image has been exported to
C:\inetpub\wwwroot\webworksheet\Release371\wwsImages\logo.jpg
with a height of 72 pixels and a width of 60 pixels.

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

Using the Background command on the Page Setup group or the 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.

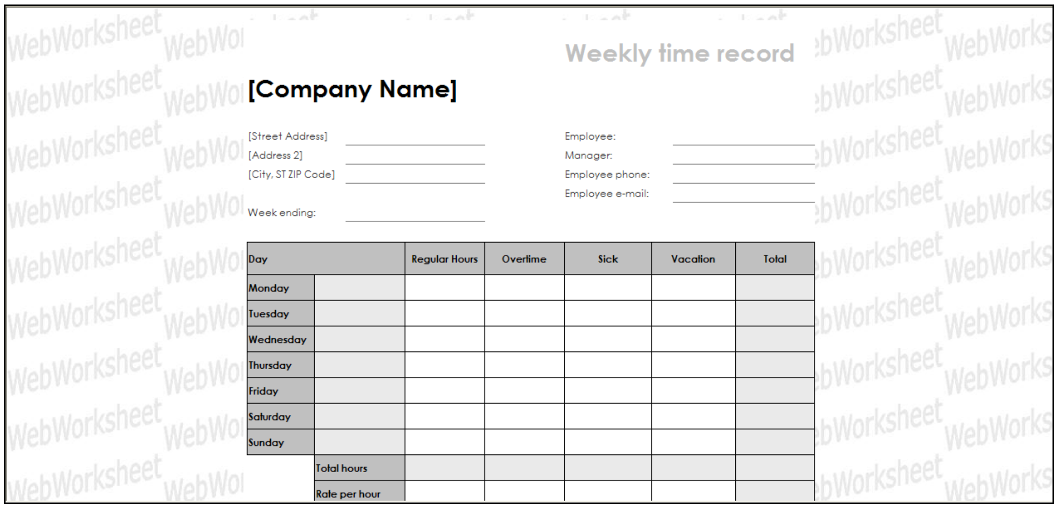
For example, placing the formula =wwsBackground("background.jpg") in cell B1



background.jpg

[Company Name]

will result in the following webworksheet:



Weekly time record

[Company Name]

[Street Address]
[Address 2]
[City, ST ZIP Code]

Week ending:

Employee:
Manager:
Employee phone:
Employee e-mail:

Day		Regular Hours	Overtime	Sick	Vacation	Total
Monday						
Tuesday						
Wednesday						
Thursday						
Friday						
Saturday						
Sunday						
	Total hours					
	Rate per hour					

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 methods which best meet 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, or stored in a database for later retrieval. 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 that are sent via email cannot be modified by the recipient.

The wwsClearButton() function restores some or all the fields to their original values.

Forms can also be saved on the user's device 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.

If your worksheet uses the "shared" or "unique" save methods, the file "webworksheetFile.php" must also be copied to your website in the same folder as the .htm file. This file can be found in the WebWorksheet installation folder on your computer.

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, cell reference, or simple formula.

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.

attributes(optional) – Additional attributes that may be applied to this input cell. See http://www.w3schools.com/tags/tag_input.asp for available attributes.

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)

=wwsInput(0, , "disabled")

=wwsInput("", FALSE, "maxlength=5")

=wwsInput(,, "autocomplete=off")

Tip: Set up the validation rules for the cell prior to entering the wwsInput function, otherwise Excel may 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

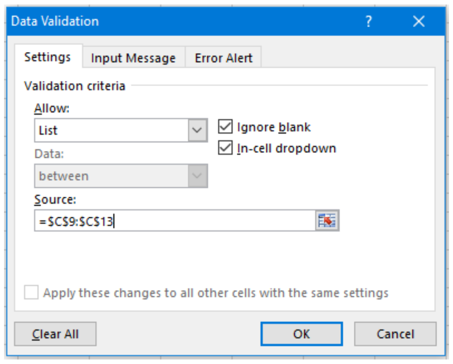
The options in the list may be defined using standard Microsoft Excel list validation functions (see http://support.microsoft.com/kb/211485). When using the validation rules to define the list of options, the list can be specified as either a cell range or a comma-delimited list. See Appendix C for details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

Examples

=wwsDropDown("Yes,No,Maybe")

=wwsDropDown(A1:B5)

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


Data Validation

Settings | Input Message | Error Alert

Validation criteria

Allow: List

☑ Ignore blank

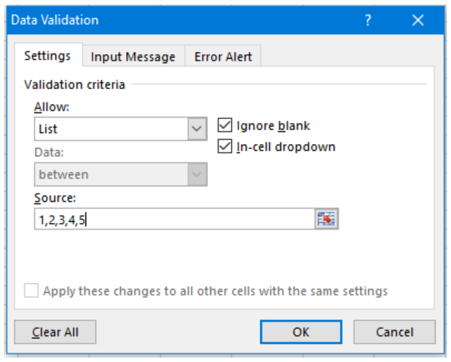
☑ In-cell dropdown

Data: between

Source: =$C$9:$C$13

☐ Apply these changes to all other cells with the same settings

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


Data Validation

Settings | Input Message | Error Alert

Validation criteria

Allow: List

☑ Ignore blank

☑ In-cell dropdown

Data: between

Source: 1,2,3,4,5

☐ Apply these changes to all other cells with the same settings

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, cell reference, or formula to be placed next to the checkbox.

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. Setting the group to -1 makes it independent of all other checkboxes.

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

Validation Options

none

Examples

=wwsCheckBox("Freight Included")

=wwsCheckBox("Blue", 1, TRUE)

=wwsCheckBox(C22, -1)

Assigning a group:

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

Tip: See Appendix B for information on required a checkbox to be selected before allowing email to be sent.

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 inside 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

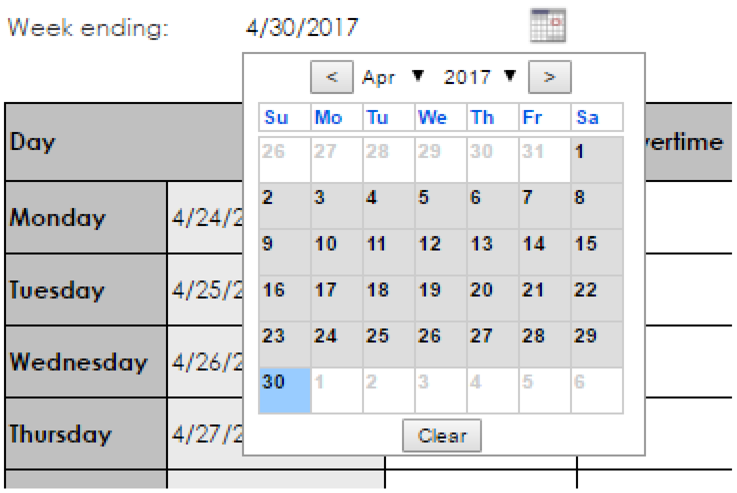
=wwsCalendar()

=wwsCalendar(TODAY())

=wwsCalendar("7/4/2017", 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 right-justified inside the input cell and will be sized to the height of the row, so the column width needs to accommodate both the date and the icon. Please note this positioning is different than earlier versions of WebWorksheet.

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 that restores cells to their previous values. Previous values are set when the form was initially created or when saved to the local device.

Arguments

button_label (required) – Text string defining the text inside the button.

clear_range (optional) – Range or text string defining which cells will be restored to their original values. If not provided, all input cells are restored.

use_local (optional) – True or False. If True, any values previously stored on the user's local device (using the wwsSaveLocal, wwsSaveButton, or wwsSubmitButton functions) will be utilized. If False, any values stored on the user's device are ignored and cells are restored to the values defined when the form was created. If not provided, it defaults to True.

Examples

=wwsClearButton(" Clear ")

=wwsClearButton("Start Over")

=wwsClearButton("New Order", A1:G10, TRUE)

=wwsClearButton("Reset", "A1:A5,B1,B5,C1:C5", FALSE)

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

Tip: To clear a non-contiguous range of cells (such as that shown in the fourth example), enclose the range within quotes.

	clear_range is not provided or clear_range = "ALL"	**clear_range = <defined_range>**
use_local = false	All input cells are returned to their default values assigned when the form was created. All values stored on the user device are deleted.	Only input cells in the defined range are returned to their default values assigned when the form was created. All values stored on the user device are deleted.
use_local is not provided or use_local = true	All input cells are returned to their last saved values on the user device.	Only input cells in the defined range are returned to their last saved values on the user device.

wwsSubmitButton

Description

Creates a button on the page that sends the completed webworksheet to one or more recipients 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 that defines the text inside the button.

email_receiver (required) – Text string or cell reference containing the email address of the recipients of the worksheet (i.e. TO). Multiple addresses must be separated with a semicolon (;).

email_sender (required) – Text string or cell reference containing the email address of the sender (i.e. FROM).

email_subject (required) - Text string or cell reference containing the subject line of the email (i.e. SUBJECT).

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.

save_range (optional) – "All", "None", or a range of cells to be saved locally on the user's device. These cell values will be remembered and displayed when the user returns to the webworksheet at a later time. If not provided, it defaults to None.

Examples

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

=wwsSubmitButton("Send", "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.",,A1:F12)
```

Tip: Pay special attention to the order of the arguments and use "" as placeholders for optional arguments when appropriate.

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 with cc: or bcc: addresses, 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.

Tip: Setting either the email_receiver or attachment_name to "@print" will display the formatted email in a new window without actually sending it so you can see the distribution list and the email body while testing.

Tip: Setting the save_range is a great way to remember user-provided data, such as email addresses or shipping addresses, when a user returns to your web page.

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 that saves the webworksheet data. See Appendix D for more details on the different ways to save data.

Arguments

button_label (required) – Text string defining the text inside the button.

method (optional) – Text string containing "local", "shared", "unique", "mysql_byCell", "mysql_byForm", "mysql_byFormWithHtml", or "mysql_asHtml" which defines how the cell values will be stored. If not provided, it defaults to "local".

password (optional) – Text string which defines the password required to save the data. Does not apply when method is "local".

location (optional) – For "shared" and "unique" methods, a text string which defines the folder where the data is stored on your website. If not provided, the data is stored in the same folder as the .htm file. When storing to a MySQL database, the cell which contains the unique record ID.

save_range (optional) – Text string which defines the range of input cells to be saved, or use "ALL" to save all the input cells. If not specified, or defined as "", all input cells are saved.

user_message (optional) – Text string which is displayed to the user after a successful save.

next_page (optional) – Text string which defines the URL of the page to be displayed after a successful save. If not provided or defined as "", the user remains on the same page.

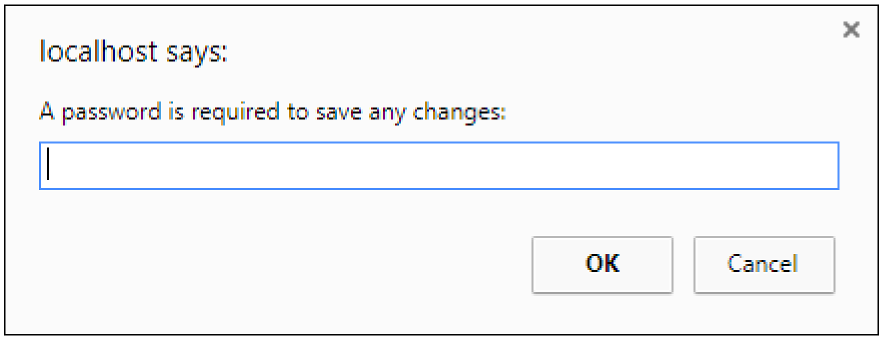
table_name (optional) – Text string which defines the database table name in which to store the data. If not provided or defined as "", the table name will be the worksheet name.

Examples

```
=wwsSaveButton("   Save   ")
=wwsSaveButton("   Save   ", "local")
=wwsSaveButton("Update", "shared")
=wwsSaveButton("Update", "shared", "P@ssw3rd!", "mydatafolder")
=wwsSaveButton("Save",  "mysql_byform", "","C2", "all", "Thanks. Your order number is " & C2 & ". \\nFind out more at www.webworksheet.com.")
=wwsSaveButton("Order","mysql_byformwithhtml","",B1,"all","Thanks. Your purchase order number is "&B1&". \\nFind out more at www.mywebsite.com.")
```

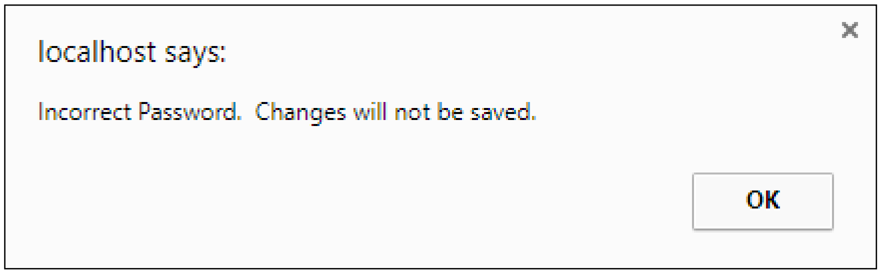
When saving the data on your webserver using either the "shared" or "unique" methods, the folder that will contain the data file must be given write access by your website administrator. In addition, the file "webworksheetFile.php", which can be found in the same folder as the .xla file, must be copied into that folder.

If a password is required to save the shared file, the user will be prompted to enter the password with a popup screen:


localhost says:

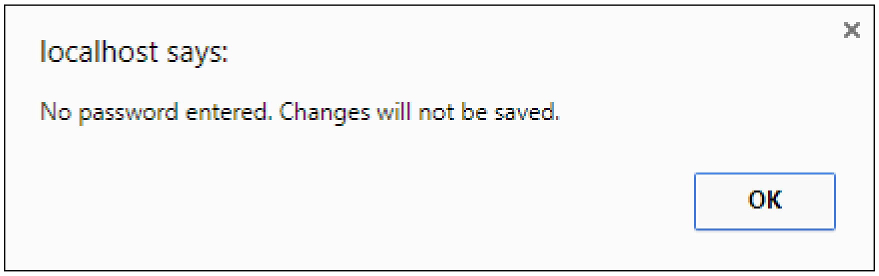
A password is required to save any changes:

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


localhost says:

Incorrect Password. Changes will not be saved.

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


localhost says:

No password entered. Changes will not be saved.

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.

Tip: Create a new folder on your website for saving the shared data. That folder must be given write access for all users, and is more secure than giving write access to the folder containing the .htm file.

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 that defines the text inside the button.

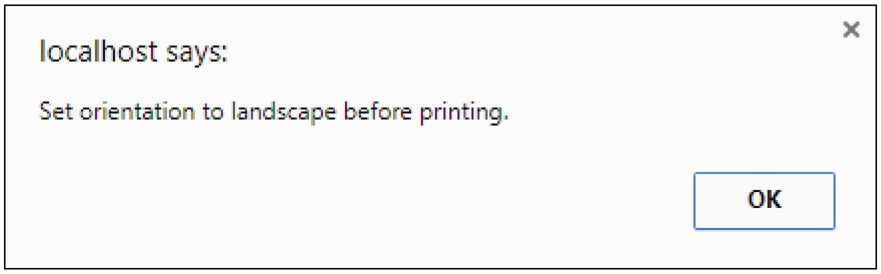
userText (optional) – Text message which can be displayed to the user prior to the print dialog box appearing.

Examples

```
=wwsPrintButton("Print")

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

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



localhost says:

Set orientation to landscape before printing.

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 that defines the text inside the button.

calculation_range (optional) – Range of cells over which the formulas will be evaluated. If not provided, 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 Monthly Payment", D5, B5)

=wwsCalculateButton("Recalc", ,namedcell)
```

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 that 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 that defines the text inside the button.

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

format_cell (optional) – cell whose format (e.g. color and font) is used to style the button.

Examples

```
=wwsCodeButton("Sort", "myCustomSort()")
=wwsCodeButton("   Sort   ", "sortView(3)")
=wwsCodeButton("Add Row", "addRow()", C5)
```

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 that 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

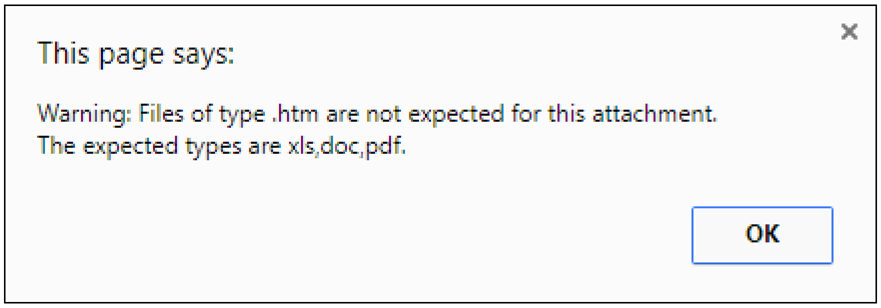
=wwsFileAttach()

=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.



This page says:

Warning: Files of type .htm are not expected for this attachment.
The expected types are xls,doc,pdf.

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"), 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"), wwsSetCell("G20", Now()), "")

Tip: This function provides the ability to change the form prior to submitting, saving, or printing it. For example, it can be used to 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 wwsUserClicked 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"), 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 MySQL database using a SQL statement and places the result in a range of cells.

Arguments

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

destinationRange (required) – Text string defining 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 and Appendix D 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.

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

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

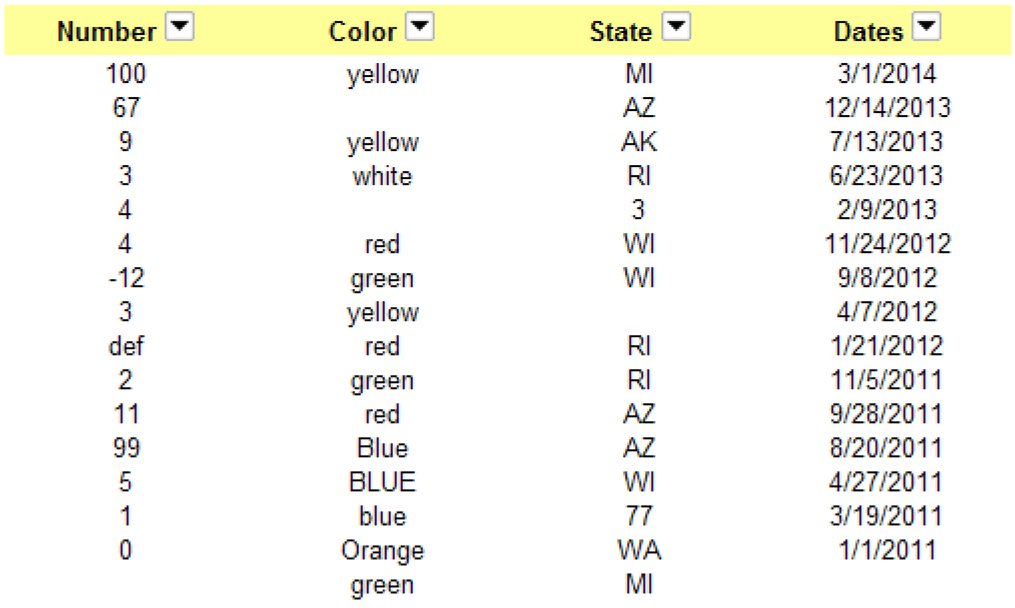
=wwsFilter("State",B4:F119)

=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*

wwsSaveLocal

Description

Saves selected cell values to the user's local device. If the browser supports local storage (as most modern browsers do), the data is stored there. If not supported, the data is stored in a cookie. See Appendix D for more information about saving data to the local device.

Arguments

saveRange (required) – Range or text string that defines the cells to be saved.

userMessage (optional) – Text string of the message to display to the user after the values have been saved.

Examples

=IF(B12<>"", wwsSaveLocal("B12"), "")

=IF(wwsOnChange(C27:C35), wwsSaveLocal("C27:C29,E33", ""), "")

=IF(wwsUserClicked("Save") = TRUE, wwsSaveLocal(C27:C35, "Your data has been saved."),"")

Tip: Use this function to remember some of the data entered by a user, such as name or address, so it is automatically displayed when a user returns to your page.

Tip: You only need to specify input cells for saving. All other cells will be re-calculated as necessary when the page is re-loaded.

Tip: Use a single wwsSaveLocal function to save all the desired cells, instead of a separate function for each cell. Each save overwrites any previous saves, so if you use multiple function calls, only the last set of cells will be saved.

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)

=wwsImage("http://www.webworksheet.com/examples/wwsImages/poweredBy.jpg",
  76, 199)
```

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.

Tip: If you want this image to appear in forms that are emailed, use the full URL of the image as shown in the last example. If your site uses certificates, preface the URL with https://.

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.

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.

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 3rd 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))

=IF(B12>B10, wwsShowRows(13) & wwsShowRows(15) & wwsHideRows(18,20), "")

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.

Tip: Multiple show and hide functions can be executed together by appending the functions with the ampersand (&) as shown in the example above.

wwsShowAndHide

Description

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

Arguments

label (required/omitted) – Text displayed in the cell. Required when used as a primary function, and omitted when used inside an IF statement.

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.

showFirst (optional) – Boolean used to define if the rows are made visible first and then hidden. If missing or set to TRUE, the showRows will be made visible and then the hideRows are hidden. If FALSE, the hideRows are hidden before the showRows are made visible.

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")
=wwsShowAndHide("Click to see shipping address", 13, 15, 10, 20, , FALSE)
=IF(OR(wwsUserClicked("Submit")=TRUE,wwsUserClicked("Print")=TRUE),
  wwsShowAndHide(ROW(A5),ROW(A69), ROW(A1), ROW(A4)),"")
```

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

Tip: Set the showFirst argument to FALSE to hide a block of rows and show just a few within that block.

Tip: The Label argument must be omitted when used as true or false clause with an IF statement, as shown in the last example.

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. This function can be placed in any cell in the row, as long as it is within the #end marker.

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 currently visible to the user 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.

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:

MyWebPage.htm#Example3

wwsActiveBorder

Description

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:

www.w3schools.com/css/css_border.asp

<*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:

www.w3schools.com/cssref/css_colornames.asp

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.

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". This must be a string and cannot be a cell reference or a defined name.

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

Description

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.

wwsOnChange

Description

Returns TRUE any time one of the cells in the specified range is changed.

Arguments

aRange (required) – Range of cells to monitor.

Examples

```
=wwsOnChange(A1:C15)
=wwsOnChange((A1:B5,C11,D4,D9:E9))
=IF(wwsOnChange(C27:C35), wwsSaveLocal("C27:C29,E33", ""), "")
```

Tip: This function is intended to be used to trigger other events, such as wwsSaveLocal. Please note that non-contiguous ranges (such as the second example), must be enclosed within parenthesis.

wwsSetup

Description

Provides a method for changing the default settings for WebWorksheet and providing configuration data.

Available Settings

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.

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.

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.

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. Setting iterations to a negative number will force ALL formulas to be evaluated every iteration without regard to their dependencies. Again, use this only if the some cells are not calculating correctly as it increases execution time.

metaTags – String defining the name of a text file to embed within the header of the generated HTML file. This file could contain any meta tags you define to allow your page to be properly indexed via a search engine.

mysql – String defining the database server name, user account, user password, and database name to connect to a MySQL database. This information is encrypted to prevent users from seeing this information. (See Appendix D for more information)

useHTTPS – Boolean (true or false) which instructs WebWorksheet to reference required files using SSL. Defaults to false if not specified. Set this to true only if your website has a digital certificate installed.

zoom – Integer defining the scaling for the generated page. If set, the zoom value in Excel is ignored and this number is used instead.

Examples

```
=wwsSetup("iterations", 6)

=wwsSetup("iterations", -3)

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

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

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

=wwsSetup("DBQueryScript", "webworksheetDBQuery.asp")

=wwsSetup("calculateOnInit", false)

=wwsSetup("useHTTPS", true)
```

```
=wwsSetup("metaTags", "myMetaTags.htm")
=wwsSetup("zoom", 100)
=wwsSetup("mysql", "ec2-52-33-160-169.us.amazonaws.com,joseph,p@sswerd,seeds")
```

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"). However, string comparisons done via the Lookup functions are case insensitive so no conversion functions are necessary.

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 that 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 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.

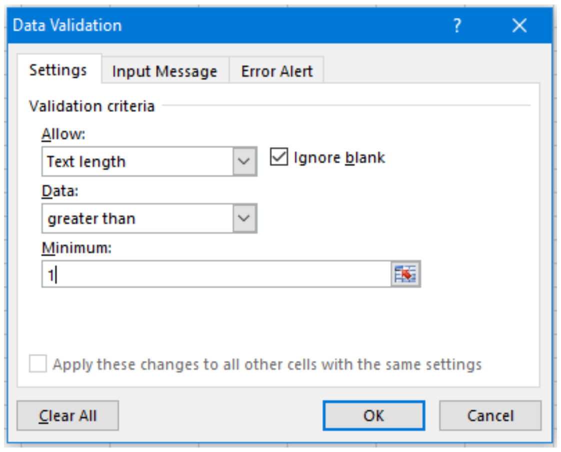
Appendix B. Validating Checkboxes before Emailing a Form

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", -1, FALSE)
```

set up the validation rule as:

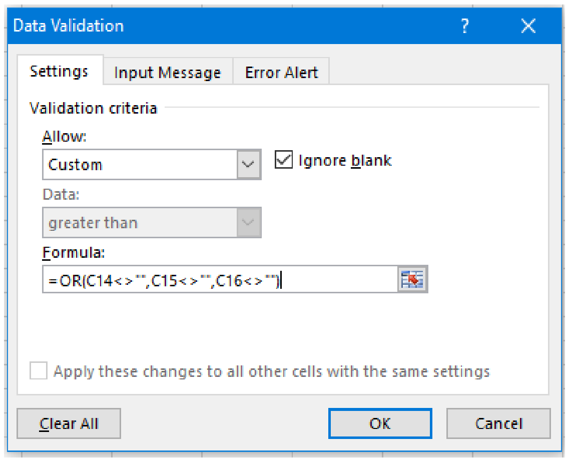


Validation criteria
Allow: Text length
[✓] Ignore blank
Data: greater than
Minimum: 1
[ ] Apply these changes to all other cells with the same settings

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", 1)
=wwsCheckbox("Neither Agree or Disagree", 1)
=wwsCheckbox("Disagree", 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.



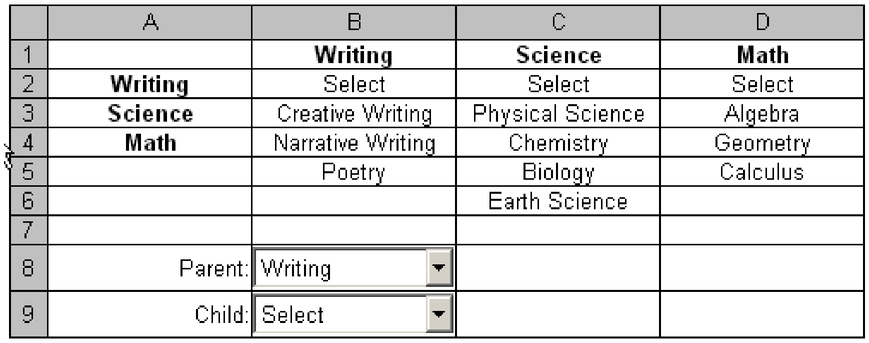
Validation criteria
Allow: Custom
[✓] Ignore blank
Data: greater than
Formula: =OR(C14<>"",C15<>"",C16<>"")
[ ] Apply these changes to all other cells with the same settings

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 ▾		

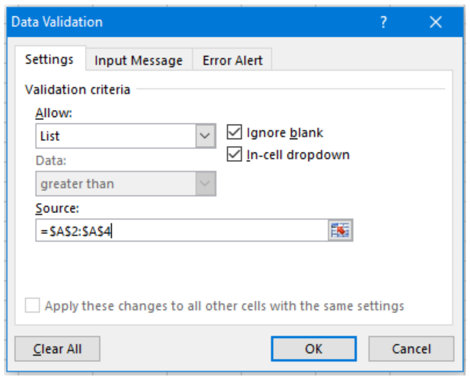
The parent options are those in Column A, with the dependents in Columns B-D. When the user selects Science in the parent dropdown, the child options are changed to those in Column C. The formula placed in the parent cell B8 is:

=wwsDropDown(A1:D6, B9)

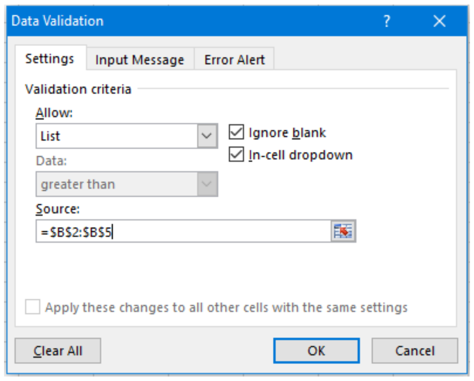
where B9 is the cell containing the child dropdown, and the formula for the child dropdown is:

=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.



Validation criteria
Allow: List
☑ Ignore blank
☑ In-cell dropdown
Data: greater than
Source: =$A$2:$A$4
☐ Apply these changes to all other cells with the same settings



Validation criteria
Allow: List
☑ Ignore blank
☑ In-cell dropdown
Data: greater than
Source: =$B$2:$B$5
☐ Apply these changes to all other cells with the same settings

If you want multiple child dropdowns from the same parent, enclose the child cells in a string:

=wwsDropDown(A1:D6,"B9,B10")

Appendix D. Saving WebWorksheet Data

WebWorksheet provides several methods for saving data entered by a website user, and that data can be stored on the user's device or on your webserver.

Saving Data Locally on the User's Device
When the save method is set to "**local**" in the wwsSaveButton function, the values of all the non-blank* cells are saved in a file on the device, typically referred to as a "cookie". Each time the HTML file is opened, it checks for the existence of that cookie, and if found, sets the value of input cells to their saved values. This allows values to be saved across visits to your website, and can be used to remember user-specific information, such as name, address, or last order quantity. Each webpage gets its own cookie, and values cannot be shared across pages.

How the data is stored, and the quantity of data that can be stored, varies across browsers and their versions. For example, versions 6 and earlier of Internet Explorer are limited to about 4000 characters per cookie. Newer versions allow much greater capacity, but if your users are using very old browsers, some data may not be remembered. Keep this in mind when defining the save range and use a defined range, instead of "ALL", to minimize the amount of data. Data stored in cookies remains on the device until the user intentionally removes those cookies, or it is deleted using the wwsClearButton function.

New browsers also provide an additional storage technique, referred to as "local storage", which may allow up to 5MB of data. Data stored in local storage remains on the device until the cache is cleared on the browser.

WebWorksheet will store data in local storage if supported by the browser, and cookies if local storage is not supported. In both cases, the user's browser must be configured to allow the use of local storage or cookies, and those are typically enabled by default.

Keep in mind that values stored locally on a device are unique to the browser and the device. Values are not shared across browsers, nor are they shared across devices. For example, if a user visits your webpage using IE and returns later using Chrome, they will not see the saved values from IE. If they visit your page using a laptop and return using their phone, again, they will not see the saved values from the laptop.

** Non-blank cells are defined as those which do or may contain an actual value, such as a text string, a formula, or may receive user input. It does not include cells which are used solely for spacing and layout.*

Saving Data on Your Website in a Shared File
Another feature of WebWorksheet allows user data to be saved on your webserver, allowing that data to be remembered across browsers, devices, and users.

When the save method is set to "**shared**" in the wwsSaveButton function, the values of all the non-blank* cells are saved in a file on your webserver. Any changes made by users are visible to all users, and is designed to allow multiple users to edit a single instance of your webpage. Each time the HTML file is opened, it checks for the existence of that file, and if found, sets the value of the input cells to their saved values. This allows values to be saved across visits and across users to your website. Each webpage gets its own file, and values cannot be shared across pages. There is no limit on the amount of data that can be saved, and the data remains on the server until manually removed.

This shared file is created on your webserver, in the folder specified in the "location" argument, and is named after the HTML file, but with a .wws extension. So if your HTML file was called seed_order.htm, the data file will be named seed_order.wws.

In order to save data files on your webserver, users must be give "write" access to the folder where the data files are stored. This is typically done by the website administrator. It is recommended that a subfolder be created below where your HTML files reside, and write access be given only to that subfolder. This would prevent an overly ambitious user from overwriting your original HTML files. The file called "webworksheetFile.php" must also be copied into the same folder as the HTML file, and can be found in the same installation folder as the .xla file.

Keep in mind that, because multiple users can edit the same page simultaneously, there is a chance that one user's changes can be overwritten by another user. If users A and B open the page simultaneously, user A saves their changes first, then user B, only user B's changes will be remembered. There is no locking mechanism or checkin/checkout feature to prevent overwriting of data.

Saving Data on Your Website in a Unique File
When the save method is set to "**unique**" in the wwsSaveButton function, the values of all the non-blank* cells are saved in a unique file on your webserver, whose name is based on a value passed on the URL. Each time the HTML file is opened, it checks for the existence of that file, and if found, sets the value of the input cells to their saved values. This allows values to be saved across visits independent of browser and device. Multiple data files may be created for a single page, and visibility to those files is controlled by dissemination of the URL parameter. There is no limit on the amount of data that can be saved, and the data remains on the server until manually removed.

For example, if you had an order form, and you wanted each customer to see only their last order form, you could give each customer a unique identifier (e.g. customer number or email address) and pass that customer number on the URL for your page. The unique identifier is passed using the "_uid_" value on the URL, such as:

```
http://www.mysite.com/orderform.htm?_uid_=8399
http://www.mysite.com/orderform.htm?_uid_=smith43
http://www.mysite.com/orderform.htm?_uid_=msmith@gmail.com
```

This unique file is created on your webserver, in the folder specified in the "location" argument, and is named using the _uid_ argument. For the first example, the data will be saved in a file named "8399.wws", and in the second example a file named "smith43.wws".

Similar to shared files, users must be give "write" access to the directory where the data files are stored. This is typically done by the website administrator. It is recommended that a subfolder be created where your HTML files reside, and write access be given only to that subfolder. This would prevent an overly ambitious user from overwriting your original HTML files The file called "webworksheetFile.php" must also be copied into the same folder as the HTML file, and can be found in the same installation folder as the .xla file.

Saving Data on Your Website in a MySQL Database
Another feature of WebWorksheet allows user data to be saved in a MySQL database on your webserver. MySQL is an open-source database that most web hosting companies provide free to their customers. In order to save your data in a database, you must first create your instance of the database using the tools provided by your hosting provider. Then you must provide the connection information to WebWorksheet by inserting the wwsSetup() function in one of the blank cells, where this setup function is defined as:

```
=wwsSetup("mysql", "<hostname>,<useraccount>,<password>,<databasename>, showErrors")
```

where

<hostname> is the machine name hosting the database
<useraccount> is user account created which has authority to create tables and rows
<password> is the password for the user account
<databasename> is the name of your database instance in MySQL
showErrors is a flag to turn on detailed error reporting

Examples:

```
=wwsSetup("mysql", "ec2-52-33-160-169.us-2.amazonaws.com,dbuser1,myp@sswerd,seed_orders")
=wwsSetup("mysql", "mysite.godaddy.com,johndoe,mydogsname,seed_orders,showErrors")
```

You MUST enter values for each of the first 4 fields. Please note that all 5 parameters are stored as a single string whose values are separated by commas. You may need to contact your hosting provider to get some of this information. This information is encrypted in the HTML file created by WebWorksheet so you don't need to worry about exposing any of your password information.

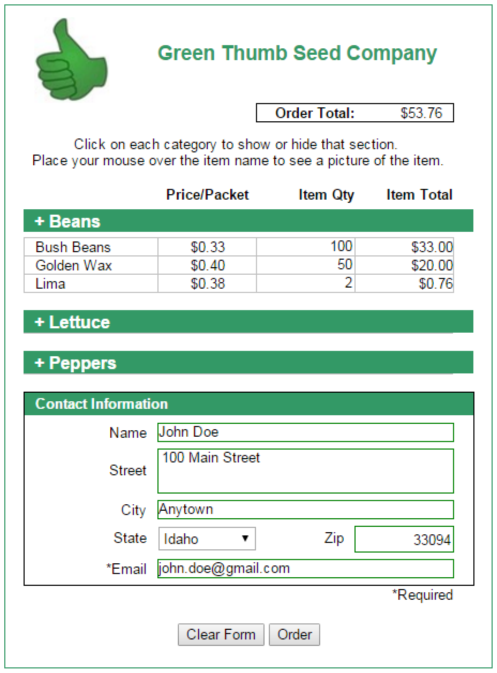
When present, the "showErrors" flag allows detailed information to be displayed if any database errors are detected, and should only be turned on while developing and debugging your worksheet. These error messages can expose information about the internal structure of your database and can be used to compromise your data. Once your worksheet is working properly, remove this flag and re-generate the HTML file.

Tip: We have set up a page to test your database connection information at

http://www.webworksheet.com/mysql_connection_test.php

There are currently four different methods for saving your data in the database: *byForm*, *byFormWithHtml*, *byCell*, and *asHtml*. Use the *byForm* method when you want the data saved in summary format, one row per form. Use the *byFormWithHtml* method when you want the data saved in summary format, one row per form, and a HTML copy of the completed form. Use the *byCell* method when you want the data saved in detail format, one row per cell. Use the *asHtml* method when you want to store only the completed form as a single entity in HTML format. Let's explain the difference in more detail.

Let's say you created an online order form, and you want the order contents to be stored in the database with each order as a separate and unique record (row) in the database. Information collected on the form, such as customer name, address, quantity ordered, and total, are to be stored in separate columns in the database table. Here's an example of a completed form:


Green Thumb Seed Company

Order Total: $53.76

Click on each category to show or hide that section.
Place your mouse over the item name to see a picture of the item.

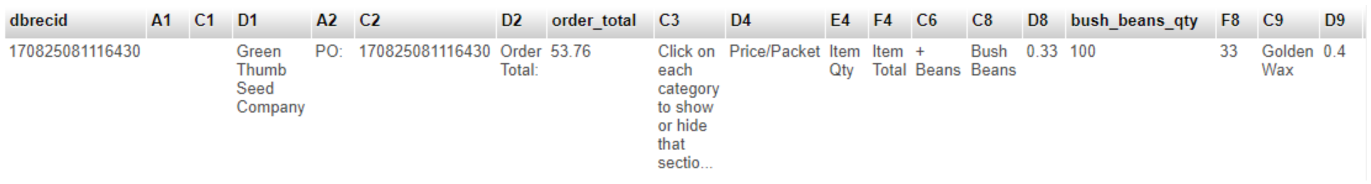
	Price/Packet	Item Qty	Item Total
+ Beans			
Bush Beans	$0.33	100	$33.00
Golden Wax	$0.40	50	$20.00
Lima	$0.38	2	$0.76
+ Lettuce			
+ Peppers			

Contact Information

Name	John Doe
Street	100 Main Street
City	Anytown
State	Idaho
Zip	33094
*Email	john.doe@gmail.com

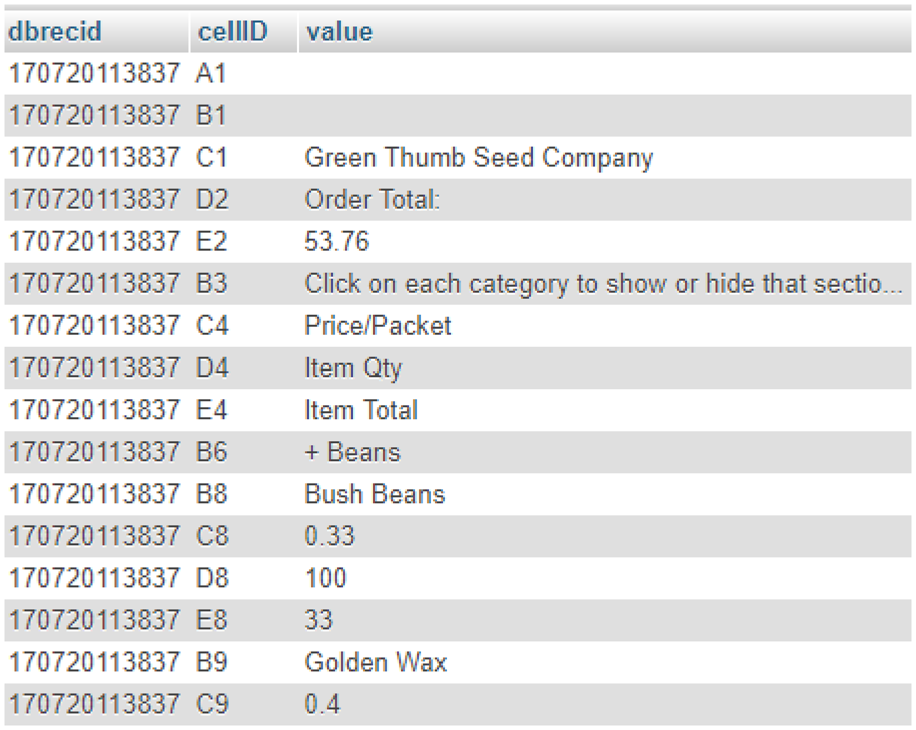
*Required

Using the *byForm* and *byFormWithHtml* methods would create database entries that would look something like this:



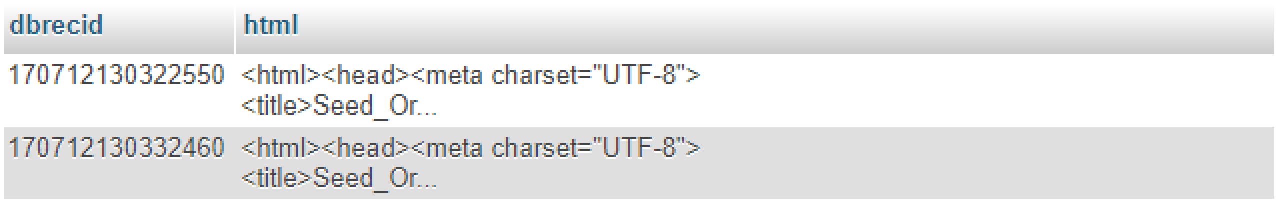
dbrecid	A1	C1	D1	A2	C2	D2	order_total	C3	D4	E4	F4	C6	C8	D8	bush_beans_qty	F8	C9	D9
170825081116430			Green Thumb Seed Company	PO:	170825081116430	Order Total:	53.76	Click on each category to show or hide that sectio...	Price/Packet	Item Qty	Item Total	+ Beans	Bush Beans	0.33	100	33	Golden Wax	0.4

Using the *byCell* method would create database entries that would look something like this:



dbrecid	cellID	value
170720113837	A1	
170720113837	B1	
170720113837	C1	Green Thumb Seed Company
170720113837	D2	Order Total:
170720113837	E2	53.76
170720113837	B3	Click on each category to show or hide that sectio...
170720113837	C4	Price/Packet
170720113837	D4	Item Qty
170720113837	E4	Item Total
170720113837	B6	+ Beans
170720113837	B8	Bush Beans
170720113837	C8	0.33
170720113837	D8	100
170720113837	E8	33
170720113837	B9	Golden Wax
170720113837	C9	0.4

Using the *asHtml* method would create database entries that would look something like this:



dbrecid	html
170712130322550	<html><head><meta charset="UTF-8"><title>Seed_Or...
170712130332460	<html><head><meta charset="UTF-8"><title>Seed_Or...

Note: the "html" column contains the html code for the entire form and can be quite large. Only a small segment of that value is shown here.

When would you use each type? In most cases, you probably want to use the *byForm* or *byFormWithHtml* methods since it's much more readable, and results in a single record created each time the save button is clicked. If you need to transfer the data to another system for processing, or want to load the saved data back into a blank form to recreate a copy of the original webpage, you would use *byCell.* Use the *asHtml* method when you only need to save an image of the completed form.

Table Names

When data is saved the first time, a new table is created in your database, whose name is derived either from the name you provided in the function call or from the name of the HTML file. Remember that the HTML name is derived from the Excel worksheet name, with spaces and other characters replaced by underscores. So if your worksheet name was "Seed Orders", the HTML file is called "Seed_Orders.htm", and the created table is named "seed_orders". Note that we automatically convert table names to all lowercase for MySQL compatibility.

Column Names

When saving data using the *byForm* and *byFormWithHtml* methods, the columns are named according to the cell address or cell name. If the cell has a defined named in Excel, such as "order_total", it will be used, otherwise the cell address (e.g. D4) will be used.

The order of the columns is determined by the order defined in the saveRange argument, unless the saveRange is defined as "" or "ALL". In those cases, the order is defined as upper-left to bottom-right. MySQL has a maximum limit of 4096 columns, so keep that in mind when defining the save range.

Please note that new columns cannot be added, or defined names created in Excel, to an existing table saved using *byForm* or *byFormWithHtml*. Since the table is created the first time a record is saved, the column names are fixed, and subsequent saves to that table with different column names (cells or defined names) will result in an error. If your form changes, you will need to give the new form a different name and save the data in a different table.

Tip: Append a version number or release date to your Excel worksheet name. Then your database can store different versions in different tables (e.g. seed_orders_v1 or seed_orders_03Jan16).

Tip: If you want to keep the same table name after making a change, you will need to "drop" the table from the MySQL database (using tools provided by your website hosting provider) before saving your form. The table will get recreated during the first save with the new changes.

Record Key
You will notice a column above called "dbrecid". In order to save data in the database and be able to uniquely identify it, it needs to be assigned a unique number called the database record id (dbrecid). You can define the dbrecid in a cell and pass that cell in the "location" argument in the wwsSaveButton function. The simplest method for defining a unique number is using a timestamp, down to the millisecond, using the following formula:

```
=SUBSTITUTE(TEXT(NOW(),"yymmddhhmmss.000"),".","")
```

You could make this more unique by appending a random number, customer number, or part of the email address if desired. But keep in mind that the dbrecid must be unique, so if you use something as simple as just the customer number, each save will overwrite the previous saved values and only the latest data would be saved.

Tip: If you want the dbrecid to be automatically assigned using the timestamp shown above, simply put "" in the wwsSaveButton function as the location argument.

Tip: Do NOT give any of the cells in your worksheet a defined name of 'dbrecid'. It will cause an error due to duplicate column names.

Data Types
When data is stored in the database table, all cell values are converted to strings regardless of the data type defined in Excel. This is done to maximize compatibility with other systems and allow flexibility in data retrieval. In MySQL parlance, the data type assigned to all columns is *text*, which has a maximum length of 64Kb. The only exceptions to this are the "dbrecid", which is stored as *varchar(255)*, and the "html" column used in the *byFormWithHtml* and *asHtml* methods, which is stored as *mediumtext*, and has a maximum length of 16Mb.

Viewing a Completed Form
If the *byFormWithHtml* or *asHtml* methods are used, a database column is created called "html" that contains an image of the completed form in HTML format. All formulas and input cells are replaced with their current values, so the form is static and cannot be changed.

To view this HTML copy, use the wwsViewHTML() function inside a hyperlink function, as follows:

```
=HYPERLINK("javascript:wwsViewHtml('<tablename>','" & <dbrecid> & "')",<dbrecid>)
```

where <tablename> is the name of the database table and <dbrecid> is the cell that contains the dbrecid. When clicked, a new window will open that displays the HTML image.

Example:

```
=HYPERLINK("javascript:wwsViewHtml('seed_orders','" & H6 & "')",H6)
```

Using Multiple Save Methods
Perhaps you want to save some information locally on the device, such as name and address, as well as saving it to a file or database. That is possible by calling a second save function using the wwsUserClicked() function. For example, to save the information locally whenever the data is saved to the database, you would place the wwsSaveButton() function in the desired cell, and then place the following formula in another cell:

```
=IF(wwsUserClicked("Save"), wwsSaveLocal("C27:C35", ""),"")
```

If you want to perform multiple actions after a button click, list all the actions in the true clause of the IF statement appended with the plus (+) sign. For example, to force rows 1-37 to be visible, remember the values in C27:E35 on the user's device, and create a html copy of the form in a table called "htmlcopies", place the following formula in a cell:

```
=IF(wwsUserClicked("Order"),wwsShowRows(ROW(A1),ROW(A37))+wwsSaveLocal(C27:E35)+
wwsSaveButton("",  "mysql_asHtml", "",B38,"all","","","htmlcopies"),"")
```

Remember that the actions defined in the wwsUserClicked() function are executed before the action associated with the button. So in the previous example, let's assume the Order button was created using the wwsSubmitButton() function used to send email. When clicked, the rows are made visible, values stored in a cookie, and the form copy stored in the database BEFORE the email is sent.

Saving Data Locally Without a Button

There may be situations where you would like to remember the user's latest entries, without requiring them to click on a save button. For example, a mortgage calculator where you want to remember the last terms they selected. This may be accomplished using the wwsOnChange() function, such as:

```
=IF(wwsOnChange(C2:C5), wwsSaveLocal("C2:C5", ""), "")
```

This example will automatically save any input cells in the range C2:C5 any time one of those cells is changed.

Loading Form Data back into the Excel Worksheet

Any data that has been saved on your webserver using a shared or unique file (ending with a .wws extension) can be loaded back into the Excel worksheet that was used to create the html page. To load the data, use the Import Data option on the WebWorksheet menu to select the .wws file to load.

Please Note: Loading data back into Excel will ***OVERWRITE*** *any existing cell values or formulas*! We suggest making a copy of the worksheet and loading data into the copy instead of the original file.