List of Custom Smarty Plugins

The Geo software extends the Smarty Template Engine, a 3rd party "software library", for the software's template system. You can read the smarty.net documentation to learn about the main functionality available in Smarty-based templates. The Geo software adds much more functionality on top of what is built into the Smarty template engine system, as noted below.

The plugins are divided into different sections: resources, block functions, functions, and modifiers. Each section describes how that type of plugin works.

Resources

In Smarty, a resource is something that gets a template. For instance, Smarty has a built-in *file* resource that simply retrieves the requested template from the Smarty template folder.

geo_tset resource

Added in **Version 6.0.0**. [1)]

This resource makes the ***cascading template set*** model possible. In a nutshell, it looks through all of the active template sets for the requested template, and gets the template from the first template set it can find the template in.

See How a Template Set is Chosen for more info.

Block Functions

In a Smarty template, a block function is a block of the template surrounded by special Smarty tags, something like:

```
{function_name} ... {/function_name}
```

This has a few different uses. Currently there is only one custom block function but there may be more added in future versions.

{add_footer_html}

Added in **Version 7.3.0**.

The **{add_footer_html}** block is used to move the contents inside the block to the end of the

contents for the {footer_html} tag, or to display the contents in-line if the admin settings disable using {footer_html}. The setting that affects this, is in the admin at **Design > Settings**, the setting "Use {footer_html} to delay loading of certain javascript". If that is enabled, the contents are moved to {footer_html}, if disabled the contents are displayed in-line.

Any time that you insert JS into a template in-line, if the JS uses jQuery, jQuery-UI, or any of the "built in" JS, it will need to use the {add_footer_html} block in order to work correctly when the above mentioned setting is turned on. And the JS needs to be coded such that it does not need to be in that exact location within the template, it "can be" or it could be moved to the end depending on settings.

Typical Usage:

```
{add_footer_html}
<script type="text/javascript">
//Some JS that uses something "built in" like jQuery, the gjUtil object, the
geoUtil
//object, jQuery UI, etc.
</script>
{/add_footer_html}
```

Required Parameters:

- There are no required parameters. <

Standard Optional Parameters:

- There are no standard optional parameters. <

Non-Standard Optional Parameters:[2)]

- There are no non-standard optional parameters <

Functions

In a Smarty template, a template function is a special tag that looks something like:

```
{function_name var1='value' var2='value2'}
```

See Smarty Docs for more information on what a Smarty function is.

{body_html}

The **{body_html}** is used to display the content specific for the current page being displayed. For instance, on the login page, the **{body_html}** displays the login form on the page.

This custom function behaves very similarly to the built-in **{include ...}** function, except that it uses a system template to display the contents[3)]. What template is used, depends on the current page

being viewed.

Typical Usage:

```
{body_html}
```

Required Parameters:

- There are no required parameters. <

Standard Optional Parameters:

- There are no standard optional parameters. <

Non-Standard Optional Parameters:[4)]

- **file** - **Added in Version 6.0.0**. You can over-write what file is normally used to display the body_html contents. Be careful when using this on templates that are attached to many different pages! <
- **g_type** - **Added in Version 6.0.0**. Over-write the template type (system, module, addon, external, or main_page) by specifying **g_type='type_name'** in the tag. Doing so will make it look in the alternate template type location for the template. <
- **g_resource** - **Added in Version 6.0.0**. Over-write the g_resource, which is basically the "top folder" once inside the template type folder. If you used a g_type of system, and g_resource of cart, and the file-name requested was sub_dir/display.tpl, the actual location would be system/cart/sub_dir/display.tpl. The g_resource is not always required, and not typically used for main_page templates, in which case it would be set to an empty string. <
- **assign** - **Added in Version 6.0.0**. Allows assigning the output to the variable specified. <
- Variables - Any other parameters used in the tag, will over-write the same-named body_html variables that are set by the script, much like the effect of adding extra parameters to an {include ...} tag. <

{head_html}

Added in **Version 7.3.0**

In version 7.3.0, we re-named the old {header_html} tag to be {head_html} instead. This is because the name {header_html} might make it sound like it is meant for the "header", which has become the standard way to refer to the top part of the page, rather than what it is really meant for, the <head> section of the HTML.

We just thought if the tag goes inside the <head>...</head> inside HTML, which in the default template, is in a file named head.tpl, we might want to use the name {head_html} instead of {header_html} just so it is clear.

Note that the old {header_html} will continue to work for the forseeable future. But if you are using version 7.3.0 or higher we recommend to go ahead and change any uses of {header_html} to the new name.

See {header_html} for documentation on this tag.

{header_html}

The **{header_html}** is used to display any HTML that needs to be included in the <head> section of the page, such as javascript, CSS links, etc. that are needed for the page being displayed. For instance, on the listing details page, needed javascript is added to allow the lightbox to function properly.

One question we get a lot is what template is used to display the {header_html} or what PHP file generates the contents. The answer is that there is no one single template or single spot that generates the entire contents. The contents are pooled together from different parts of the script, for instance one module might need a specific JS file to be loaded, another might need some CSS to be loaded, etc. The contents of this tag consist of many different parts that are needed for the different parts of the page being displayed. There are a few specific pages that do use a template to display the "bulk" of the content[5)], but even then most of the content is put together from across the software.

Typical Usage:[6)]

```
{header_html}
```

Required Parameters:

- There are no required parameters. <

Standard Optional Parameters:

- There are no standard optional parameters. <

Non-Standard Optional Parameters:[7)]

- **file** - **Added in Version 6.0.0**. You can over-write what file is normally used to display the header_html contents. This will only have an effect on the rare pages that use a template to display the main part of the contents. Be careful when using this on templates that are attached to many different pages! <
- **g_type** - **Added in Version 6.0.0**. Over-write the template type (system, module, addon, external, or main_page) by specifying **g_type='type_name'** in the tag. Doing so will make it look in the alternate template type location for the template. This will only have an effect on the rare pages that use a template to display the main part of the contents. <
- **g_resource** - **Added in Version 6.0.0**. Over-write the g_resource, which is basically the "top folder" once inside the template type folder. If you used a g_type of system, and g_resource of cart, and the file-name requested was sub_dir/display.tpl, the actual location would be system/cart/sub_dir/display.tpl. The g_resource is not always required, and not typically used for main_page templates, in which case it would be set to an empty string. This will only have an effect on the rare pages that use a template to display the main part of the contents. <
- **assign** - **Added in Version 6.0.0**. Allows assigning the output to the variable specified. <
- Variables - Any other parameters used in the tag, will over-write the same-named header_html variables that are set by the script, much like the effect of adding extra parameters to an {include ...} tag. <

{footer_html}

Added in **Version 7.3.0**

The **{footer_html}** is used to display "delayed" contents that do not need to be loaded until the end of the page, typically JS. This tag will only be used if the setting in the admin page **Design > Settings** named **Use {footer_html} to delay loading of certain javascript** is enabled (checked).

Normally all of the JS for the page used by the system, such as the JS that makes the lightbox work, is loaded by {head_html} inside the <head> section of the page. But if the above mentioned setting is enabled, the JS is displayed in the {footer_html} instead. This is referred to as delaying the JS loading, the "overall" page speed is not affected since it still has to load everything, but the "perceived page speed" is faster since it delays loading the JS until the end of the page. See the user manual for the setting **Use {footer_html} to delay loading of certain javascript** for more information.

So in other words, the {footer_html} is basically the parts of the {head_html} tag that can be "delayed" until the very end of the page. See the notes on {head_html} for more information.

Typical Usage:[8)]

```
{footer_html}
```

Required Parameters:

- There are no required parameters. <

Standard Optional Parameters:

- There are no standard optional parameters. <

Non-Standard Optional Parameters:[9)]

- **file** - You can over-write what file is normally used to display the header_html contents. This will only have an effect on the rare pages that use a template to display the main part of the contents. Be careful when using this on templates that are attached to many different pages! <
- **g_type** - Over-write the template type (system, module, addon, external, or main_page) by specifying **g_type='type_name'** in the tag. Doing so will make it look in the alternate template type location for the template. This will only have an effect on the rare pages that use a template to display the main part of the contents. <
- **g_resource** - Over-write the g_resource, which is basically the "top folder" once inside the template type folder. If you used a g_type of system, and g_resource of cart, and the file-name requested was sub_dir/display.tpl, the actual location would be system/cart/sub_dir/display.tpl. The g_resource is not always required, and not typically used for main_page templates, in which case it would be set to an empty string. This will only have an effect on the rare pages that use a template to display the main part of the contents. <
- **assign** - Allows assigning the output to the variable specified. <
- Variables - Any other parameters used in the tag, will over-write the same-named footer_html variables that are set by the script, much like the effect of adding extra parameters to an {include ...} tag. <

See Also: The {add_footer_html} function block.

{addon}

The **{addon ...}** tag displays the contents of a tag generated by an addon. You don't have to guess at what the exact tag should be, in the admin at **Addons > Manage Addons** when you hover over any addon, it lists the tags available for that addon. See each of the addon's documentation for the usage of each addon tag.

Typical Usage:

```
{addon author='auth_tag' addon='addon_name' tag='tag_name'}
```

Requires Attachment: Addon tags are attached to **main_page templates**[10], every time you edit a template in the admin panel, the template is re-scanned and attachments re-saved. If you edit the template outside the admin panel, in **Design > Template Sets**, use the **Re-Scan Attachments** button.

<tip c t>**Tip:** In the admin template editor, at **Design > Manager**, use the **Insert Tag** blue button. Then click on the **Addon Tags** tab within the tool, and you will see a drop-down of all the available addon tags in all of your currently enabled addons!</tip>

Required Parameters:

- **author** - This should be set to the addon's "Author Tag"[11]. <
- **addon** - The addon's folder name. <
- **tag** - The tag name for which tag to display. Addons can have multiple tags. <

Standard Optional Parameters:

- **headOnly** - **Added in Version 6.0.0** Set this parameter to 1 to make it so that only the header stuff is added, the addon tag itself is not displayed. Useful for times when you need the automatic header contents to be added to {header_html} but the addon tag is not inside a main_page template. In such cases, you can just add the addon tag to the main page template used to display that same page, and add **headerOnly=1** to the tag, and it will keep the template from actually displaying the tag in the spot inserted. <

Non-Standard Optional Parameters:[12]

- **g_type** - **Added in Version 6.0.0**. **Only used if** the addon tag uses the built-in template methods to display the tag contents. Over-write the template type (system, module, addon, external, or main_page) by specifying **g_type='type_name'** in the tag. Doing so will make it look in the alternate template type location for the template. <
- **g_resource** - **Added in Version 6.0.0**. **Only used if** the addon tag uses the built-in template methods to display the tag contents. Over-write the g_resource, which is basically the "top folder" once inside the template type folder. <
- **assign** - **Added in Version 6.0.0**. Allows assigning the output to the variable specified. <
- Other:
    - Template Variables - **Only used if** the addon tag uses the built-in template methods to display the tag contents. Any other parameters used in the tag, will over-write the same-named template variables that are set by the addon, much like the effect of adding extra parameters to an {include ...} tag. <
    - Other parameters - See the addon tag documentation. The addon tag does have available

the parameters that were used in the tag, so it is possible for a specific addon tag to use, or perhaps even "require" additional parameters. You would need to refer to the addon documentation provided by the addon's author for any information about such "additional" parameters.

{external}

The **{external ...}** tag will display the URL for the requested external file, such as an image, CSS, or JS file located in the external/ folder in a template set.

<tip c t>**Tip:** In the admin template editor, at **Design > Manager**, use the **Insert Tag** blue button. Then click on the **external** tab within the tool, and you will see a drop-down of all the external files found.</tip>

See How a Template Set is Chosen for more info on how it determines which template set to use.

Typical Usage:

```
{external file='images/background.jpg'}
```

Required Parameters:

- **file** - What file to look for in the external folder in each of the template sets. <

Standard Optional Parameters:

- There are no standard optional parameters. <

Non-Standard Optional Parameters:[13)]

- There are no non-standard optional parameters for this tag. <

{module}

The **{module ...}** tag will display the specified built-in module. You can easily find and insert any built-in module tag into a template, when using the template editor in **Design > Manage**, by using the **Insert Tag** tool. It will give a drop-down list of all the modules available, just select the one desired and click the button to insert the tag.

Typical Usage:

```
{module tag='title_module'}
```

<tip c t>**Tip:** In the admin template editor, at **Design > Manager**, use the **Insert Tag** blue button. Then click on the **module** tab within the tool, and you will see a drop-down of all the available modules! Select one to insert the tag into your template.</tip>

Requires Attachment: Addon tags are attached to **main_page templates**[14)], every time you edit a template in the admin panel, the template is re-scanned and attachments re-saved. If you edit the

template outside the admin panel, in **Design > Template Sets**, use the **Re-Scan Attachments** button.

Required Parameters:

- **tag** - The tag name used by the module you want to display. <

Standard Optional Parameters:

- There are no standard optional parameters. <

Non-Standard Optional Parameters:[15)]

- **assign** - **Added in Version 6.0.0**. Allows assigning the output to the variable specified. <
- Template Variables - Any other parameters used in the tag, will over-write the same-named module template variables that are set by that module, much like the effect of adding extra parameters to an {include ...} tag. A few of the most commonly changed template variables are listed below[16)]:
  - **browse_tpl** : Added in **Version 7.1.0**: Can use this to change what template to use to display the listings. This only works with *browsing modules*, modules that display a set of listings, such as the featured modules or the hottest listings module. Set this to **common/list_view.tpl** for list view, **common/grid_view.tpl** for the grid view, or **common/gallery_view.tpl** for the gallery view. <

<

- Module Settings - **(As of Version 7.0.2)** Any module setting, which is normally set in the admin panel, is able to be overridden by a tag parameter. See the list below for a complete list of module settings and what each one does. <

Module Settings (requires **version 7.0.2 or higher** to work)

When to Use: Note that each of these settings are normally changed in the admin panel. We only advise to use a module tag parameter to change this if the same module is **used in multiple locations**, and you wish to **use different module settings for the different locations**.

Setting	Valid Values	Module(s) Used For	Description
browse_view	Text: either 'grid', 'list', or 'gallery'	Any that display listings normally in **grid** layout[17)]	**Added in version 7.3.0**: Note that it is possible in previous versions, but was more complicated. This is like a shortcut to easily use either the grid, list, or gallery views.
module_number_of_ads_to_display	Number (like "5")	Any that display listings	Control how many listings are displayed.
module_display_header_row	1 (on) or 0 (off)	Any that display listings	Shows column headers and/or module title above

Setting	Valid Values	Module(s) Used For	Description
module_display_ad_description	1 (on) or 0 (off)	Any category navigation	Despite the name, it is on/off for whether to show the "return to ..." link below category navigation to link to the parent category.
module_display_ad_description_where	1 (on) or 0 (off)	Any that display listings in grid view	Whether to show description under title (1) or on it's own column (0)
module_file_name	Valid PHP file name in modules folder	All	Should not use! Well OK you can use it, if for instance you had your own custom module file to use instead of the built in one. Not recommended though, you probably want to make a custom addon tag instead.
module_replace_tag	DO NOT USE	All	Do Not Use! May cause rip in space-time continuum![18)]
module_display_username	Number from 1 to 6	display_username module	Change what info is displayed about the user - 1: username, 2: first name, 3: last name, 4: First name Last Name, 5: Last name First name 6: Email
module_thumb_width	Number	Any showing listings with images	Controls max thumbnail **width**. Note that it should be a number only, do not include "px" at the end.
module_thumb_height	Number	Any showing listings with images	Controls max thumbnail **height**. Note that it should be a number only, do not include "px" at the end.
module_number_of_columns	Number	Featured pic modules	Number of columns to use.
module_category_level_to_display	1 (on) or 0 (off)	Category level navigation	Switch to make it show one category below top level, sort of special case for site set up a specific way.
module_category	Number	Category navigation	The category ID to start out in for category navigation.
cat_id	Number	Any showing listings	Show only listings in specified category ID or subcategories[19)]. Added in **version 7.2.0**.

Setting	Valid Values	Module(s) Used For	Description
not_cat_id	Number	Any showing listings	Show only listings **NOT** in specified category ID or sub-categories[20]. Added in **version 7.2.0**.
module_display_new_ad_icon	1 (on) or 0 (off)	Category navigation	Whether to display the new ad icon or not.
module_display_type_listing	Number: 0, 1, 2, or 4	Any that display listings	What type of listings to show: 0: all, 1: classifieds only, 2: any auction type, 4: only reverse auctions
module_display_type_text	1 (on) or 0 (off)	Any that display listings	Whether to show the listing type icon or not
module_display_sub_category_nav_links	1 (on) or 0 (off)	Category navigation	Whether to show sub-categories or not.
photo_or_icon	0 (icon), 1 (thumbnail) or 2 (site default)	Any showing listings	Whether to show icon or thumbnail
use_pagination	1 (on) or 0 (off)	Any showing listings	If on, will add AJAX pagination to the module, allowing extra listings beyond the first set to be loaded without reloading the page

{listing}

Added in **Version 7.1.0**.

The **{listing ...}** tag displays blocks of information about a listing. It can be used in the **listing details** page in any part of the templates for that page, or in a **browsing sub-template**, the template used to display each individual listing when browsing the site. Note that this includes **browsing / featured pic / featured modules** inside the module templates as well! When used in those contexts, the system will automatically detect what listing to display the info for. Or to be more technical, as long as there is $listing array, with "id" set in the array, OR $listing_id set in the template, it will be able to detect that automatically. This allows it to be used in a great variety of places within templates and makes it a very versatile tag.

It can also be used anywhere in any template, even when not browsing or not on listing details, as long as you specify the **listing_id** parameter in the tag so that it knows what listing to use.

<tip c t>**Tip:** In the admin template editor, at **Design > Manager**, use the **Insert Tag** blue button. Then click on the **Listing** tab within the tool, and you will see a drop-down of all the available listing tags! It even shows any listing tags added by enabled addons as well!</tip>

Typical Usage examples:

- Standard listing **tag**

```
{listing tag='image_block'}
```

<

- Standard listing **field**

```
{listing field='price'}
```

<

- Standard listing **addon tag**

```
{listing addon='example' tag='listing_tag_example'}
```

<

Advanced Usage Examples:

- Assign array of listing tags to template var $these_listing_tags:

```
{listing field='tags' format='array' assign='these_listing_tags'}
```

<

- Assign the pre-currency symbol **unformatted** to the template var $precurrency_raw:

```
{listing field='precurrency' format='raw' assign='precurrency_raw'}
```

<

- Assign the category questions, and display the value for question with ID of 123[21] **(Requires at least 7.2.1)**:

```
{listing field='extra_questions' format='array' assign='questions'}
{if $questions.123}
    {* only display if question with ID 123 exists *}
    <strong>Question:</strong>  {$questions.123.question}<br />
    <strong>Value:</strong>
    {if $questions.123.value}
        {* This is a standard value *}
        {$questions.123.value}
    {elseif $questions.123.link}
        {* This is a URL question, so make it use a link *}
        <a href="{$questions.123.link}">{$questions.123.link}</a>
    {/if}
{/if}
```

<

Required Parameters: Requires **ONE** of the parameters to be defined in the tag, **not both**:

- **tag** - What tag to show. Tags typically take some data attached to the listing, and display it in a friendly way. For instance, the *image_block* will display all of the images attached to the listing, using gallery or filmstrip style to show them. <
- **field** - What listing field to display. This will be information about a listing displayed with no HTML markup "added", that does not require a sub-template to display, such as the listing title or description. <

Standard Optional Parameters:

- **addon** - Requires using **tag** parameter, does not work with field. This is used to allow an addon to display information about a listing. <
- **format** - Requires using **field** parameter. Recommended to use **assign**[22)] in conjunction with this parameter. There are only 2 valid values:
  - **format='raw'** - Gets the raw value of the field requested, in the format as stored in the database. Make sure you perform any formatting necessary depending on the field. <
  - **format='array'** - Only works with a few select fields:
    - listing[23)]<
    - tags[24)]<
    - order_items[25)]<
    - images<
    - seller<
    - high_bidder - Added in **Version 7.1.1** [26)]<
    - multi_level_fields (or "leveled" for short)<
    - regions - includes "additional" regions if used.<
    - extra_questions - Added in **Version 7.2.1** [27)]<
    - checkboxes - Added in **Version 7.2.1**<

< <

- **listing_id** - Used to force it to use the specified listing. If expired or invalid ID is given, and it cannot detect a listing ID to use based on the context/location of the tag, the tag will not display anything. This is required if using the listing in a template used on a page where the listing cannot be detected. The typical use of this would be to display information about a specific "example listing" that is perhaps set to not expire, within an extra page template, or similar location. Note that this is not commonly used, but it is here if you need it[28)].<

Non-Standard Optional Parameters:[29)]

- **file** - You can over-write what file is normally used to display the tag contents.[30)] <
- **g_type** - Over-write the template type (system, module, addon, external, or main_page) by specifying **g_type='type_name'** in the tag. Doing so will make it look in the alternate template type location for the template.[31)] <
- **g_resource** - Over-write the g_resource, which is basically the "top folder" once inside the template type folder, or the addon folder for an addon tag. If you used a g_type of system, and g_resource of cart, and the file-name requested was sub_dir/display.tpl, the actual location would be system/cart/sub_dir/display.tpl. The g_resource is not always required, and not typically used for main_page templates, in which case it would be set to an empty string.[32)] <
- **assign** - Allows assigning the output to the variable specified.[33)] <
- Variables - Any other parameters used in the tag, will over-write the same-named template variables that are set by the script for the sub-template used, much like the effect of adding extra parameters to an {include ...} tag.[34)] <

{debug_templates_used}

Added in **version 6.0.0**.

The {debug_templates_used} displays all template files referenced at all up until the point the tag is inserted. If a template is included below the tag for instance, that template would not be listed in the list. It is a great tool for designers needing to see exactly what system templates are used to build a page.

When you insert the tag, it will display tips and notes above the list, read those for important and useful info about the tag and its usage.

Special Tag: Not recommended for live sites:

This tag is made for developers and designers already familiar with the software and how Smarty templates work, to be able to quickly and easily see what templates (including system templates) are used to build a page. It is not intended for live sites.

Modifiers

Modifiers are used to manipulate a variable in some way, for instance the custom modifier **displayPrice** will change something that normally looks like *12345.67* into something that looks like *$12,345.67 USD*. See the Smarty.net docs for more information about modifiers.

displayPrice

This is used to take a number like **1234.56** and display it in price format like **$1,234.56 USD**.

Anything non-numeric will be converted to 0.

Typical Usage:

```
{$number|displayPrice}
```

Use site-wide settings for pre and post currency.

```
{$number|displayPrice:'pre':'post'}
```

Use specified pre and post currency values instead of site-wide settings.

```
{$number|displayPrice:false:false:cart}
```

Use site-wide settings for pre and post currency, and specify that the "type" of cost is cart cost, so that if cost is 0 it can potentially be replaced by value specified in admin.

Parameters:

- Parameter 1 - optional - Pre-currency string. If not specified or set to null, will use site-wide settings. <
- Parameter 2 - optional - Post-currency string. If not specified or set to null, will use site-wide settings. <
- Parameter 2 - optional - Price type, either 'cart' or 'listing'. If not specified or set to null, 0 prices will display normally with no replacements. <

escape_js

Used to escape a string to be usable inside JS. Replaces any newlines, tabs, or carriage returns with a single space, and adds slashes to escape the string.

Typical Usage:

```
{$string_var|escape_js}
```

No additional parameters

format_date

Similar to the Smarty modifier date_format, but uses the PHP function **date()** to format the date instead of strftime(). The var being modified should be a unix timestamp.

Typical Usage:

```
{$timestamp|format_date}
```

Will use the date format as set in the admin settings.

```
{$timestamp|format_date:'F j, Y, g:i a'}
```

Format the date as specified in the first parameter.

```
{$smarty.now|format_date}
```

Display the current time in the format specified in the admin panel settings.

Parameters:

- Parameter 1 - optional - The date format, should use syntax as described on PHP's date() docs. If not specified, will use date format settings set in the admin panel. <

fromDB

Decodes the a string that was previously encoded to save in the database. This is not typically needed in main_page templates, unless you are making use of "raw" data directly from the database

that has not already been "decoded" for you. One example of this is in listing details template, the array **$listing_data_raw** holds all of the listing's data "un-encoded".

Typical Usage:

```
{$listing_data_raw.description|fromDB}
```

No additional parameters

qr_code

Transforms a string (usually a URL) into a QR Code (Requires Geo v6.0.0+)

Typical Usage:

```
{$someText|qr_code}
```

Will create a QR code with the default parameters (size 100, UTF-8, error correction level 'L')

```
{$someText|qr_code:125:'UTF-8':'M'}
```

Will create a physically larger code with more error correction

Parameters:

- Parameter 1 - optional - integer (default: 100) containing the height and width (its square) of the finished image, in pixels. Note that this INCLUDES a border of whitespace around the code (part of the QR spec). Also note that images much smaller than size 75 or so are not likely to be readable by current smartphones (in testing, an iPhone4 had no problem with size 75, but didn't have enough camera resolution to read size 50)
- Parameter 2 - optional - The character encoding of the original string. This should almost always be 'UTF-8'
- Parameter 3 - optional - The error correction level. One of the following characters: 'L', 'M', 'Q', or 'H' (default 'L'), offering automatic correction of up to 7%, 15%, 25%, and 30% (respectively) of missing, misread, or obscured data, at the cost of less room for actual data in the code (redundant error checking)

1)
Earlier versions accomplished the same by extending Smarty's built in functions that retrieve the template. In Geo version 6.0.0 Smarty was updated to version 3.0, in which it was better to create a resource plugin rather than over-writing Smarty's built in functions.

2) , 4) , 7) , 9) , 12) , 13) , 15) , 29)
Using any non-standard optional parameter may require adjustments after updates. Problems caused by using these parameters are not covered by support.

3)
There are special cases, such as extra pages, listing details, and any pages created by addons.

5)
such as the media upload page

6)
Needs to be placed somewhere in the <head> section of the template.

[8]

Needs to be placed right before the end </body> tag in the template.

[10], [14]

Note that **only main_page templates** are able to have attachments. You will not be able attach module or addon tags in any other template. The only exception, is if that template is a "child" of a main_page template, and that main_page template has the tag attached but commented out. This is by design, in order to keep the template system as **fast and optimized** as possible.

[11]

Set by the **$auth_tag** variable in the info.php file for the addon.

[16]

OK for now, only one is shown, we'll add more examples if others become common.

[17]

This will not work for featured pic modules

[18]

Seriously, don't use this one. Only reason we put it here is to say not to use it.

[19], [20]

Note: setting to show sub-category listings must be enabled.

[21]

You would change the "123" in the example, to the question ID number you wish to display. The example shows the question and the value. Like the rest of the examples, this is a sample only of "one way" you can use this tag, meant to illustrate how to use it, you don't have to display it exactly like the example shows.

[22]

Although it is not required, in a lot of cases if the value is not assigned to a variable, it would simply display Array or similar.

[23]

entire array of listing info for the listing un-formatted

[24]

the listing tags/keywords.

[25]

That's right, if you need array of order item info for a listing, you can get it. This is not common however.

[26]

This will only work with auctions that have at least one bid. This will include both the bid information, and the user's information in the same array.

[27]

This will give an array in the format

```
array ($question_id => array (
    'question'=>'Question',
    'value' => 'The value (or answer)',
    'link' => 'http://example.com'
));
```

- Note that the 'link' will be the URL if the question type is set to be a link, or will be false otherwise. Also note that all the values will already be formatted for display.

[28]

So in other words, don't feel obliged to go inventing some way to use this option, it isn't common at all. Unless you really want just for the fun of it!

[30], [31], [32], [33], [34]

Note that some addon tags may not use this, as it depends on how the addon's author made the addon tag work. All addons created by Geodesic Solutions do work with this.

From:
https://geodesicsolutions.org/wiki/ - **Geodesic Solutions Community Wiki**

Permanent link:
https://geodesicsolutions.org/wiki/designers/template_error

Last update: **2016/02/01 10:32**