

Using Shortcodes to Display and Manipulate Custom PHP Code within WordPress Posts or Pages

- http://technerdia.com/579_shortcodes.html

From [Wordpress](#), a [Shortcode](#) is: a simple set of functions for creating macro codes for use in post content.

```
function apiFunc( $atts ) {
    extract( shortcode_atts( array( 'type' => "", 'category' => "", 'product' => "",
    ob_start();
    |   include('api.php');
    $output_string = ob_get_contents();
    ob_end_clean();
    |   return $output_string;
}
add_shortcode( api_data, apiFunc ); /* [api_data] */
```

Shortcodes are the perfect tool to use for pulling in Affiliate [API's](#) directly into Wordpress, by passing in variables within the Shortcode directly, which manipulates the output or display of the API. Because Shortcodes are built in Wordpress Functions, caching plugins will work on them. Another words, using Shortcodes to display API's instead of writing a custom plugin, means you don't also have to write in extra code for caching!

The functions.php file

Shortcode Functions get added directly into the *functions.php* file of a theme.

A Basic Shortcode Function

Shortcodes need a function assigned to them to make them, function. Below is a basic example of this taking place.

Place the below within the *functions.php* file:

```
function myShortcode() {  
    return 'Hello!';  
}  
add_shortcode( shortcode_name, myShortcode );
```

In the last line above, the `add_shortcode` Function contains the name of the short code (`shortcode_name`) and calls the function above it (`myShortcode`). The name of this shortcode is: `shortcode_name` and the function name is: `myShortcode`

- Use a shortcode name that relates to the purpose of the function.
- Always use lowercase letters in your Shortcode names.

Shortcode Call: The Shortcode will return the text: Hello

```
[shortcode_name]
```

This is a rather basic example, you can find many ways to use them with a quick [Google Search](#), they are amazingly versatile, so don't be afraid to play with them.

Passing an Attribute to the Shortcode

Rather than simply returning Hello, the Shortcode can have text passed directly to it. The text can be returned, or used to activate an if or switch statement. The example below checks to see if `$atts` is passed in, if so it returns the results of `param`, which is Hello World!

```
function myShortcode( $atts ) {  
    if ( $atts ) {  
        return $atts['param'];  
    }  
}  
add_shortcode( shortcode_name, myShortcode );
```

Passing in an Attribute:

```
[shortcode_name param="Hello World!"]
```

Both of the above examples are some what limited, to unlock the full potential of Shortcodes, Attributes need to be passed in.

Attributes within Shortcodes

Attributes are basically Variables (\$a) that get passed into the Shortcode Function.

This is done simply by adding in the following to the function:

```
extract( shortcode_atts( array( 'variable' => 'value', ), $atts ) );
```

The array() contains a variable(s) that may or may not equal something, in this case the variable equals value. Add more Variables by adding to the Array:

```
'variable' => 'value', 'another-var' => 'another-value',
```

Full example:

```
extract( shortcode_atts( array( 'variable' => 'value', 'another-var' => 'another-value', ), $atts ) );
```

Typically using Shortcodes with Attributes means something more complex is happening than displaying a single line of text, being so the output needs to be captured before displaying it.

Output Buffering

An Advanced Shortcode is basically useless without capturing the output of the code. Without it, code can display at wrong times or locations and depending on what the code does within the Shortcode function, it may return header warnings.

To correct this we use a few PHP functions: [ob_start](#), [ob_get_contents](#) and [ob_end_clean](#). These functions capture the output of the code into a memory first, then displays (or cleans) the output based on the order it was captured in.

Example Shortcode Function Using Output Buffering

In the example below, we capture the if statements results based on the variable's value.

```
function myShortcode( $atts ) {
    extract( shortcode_atts( array( 'variable' => "", ), $atts ) );
    ob_start();
    if ( $variable == "" ) { echo "Do Nothing!"; }
    if ( $variable == '1' ) { echo "Do This Code!"; }
    if ( $variable == '2' ) { echo "Do That Code!"; }
    $output_string = ob_get_contents();
    ob_end_clean();
    return $output_string;
}
add_shortcode( short_code_name, myShortcode);
```

This Shortcode will return: *Do This Code!*

```
[shortcode_name variable="1"]
```

Advanced Example Shortcode Function

This example captures an XML Feed, loads it into a string, then manipulates the output based on the passed in Attributes. This Shortcode can display various outputs based on the value of type. When type=cars a unique API Url is called, then we foreach through each result, correct the HTML entities and cut the description length off at 255 characters before returning them.

```
function myShortcode( $atts ) {
    extract( shortcode_atts( array( 'type' => "", 'debug' => "", ), $atts ) );
    ob_start();
    /* start code */

    if ( !$type ) { $type = "cars"; }

    if ( $type == "cars" ) {
        $apiUrl = file_get_contents("http://apiurl.com/");
    }

    if ( $type == "boats" ) {
        $apiUrl = file_get_contents("http://apiurl.com/");
    }

    if ( $type ) {
        $xml = simplexml_load_string($apiUrl);
        if( $debug ){echo "<pre>";print_r($xml);echo "</pre>"; exit;}
    }

    if ( $type == "cars" ) {
        foreach($xml->cars as $car){
            $title = $car->title;
            $about = $car->description;
            echo '<h2>'. htmlentities($title); .</h2>';
            echo '<p>'. substr($about,0,255); .</p>';
        }
    }

    if ( $type == "boats" ) {
        $name = $xml->boats->boat->name;
        echo $name;
    }

    /* end code */
    $output_string = ob_get_contents();
    ob_end_clean();
    return $output_string;
}
add_shortcode( short_code_name, myShortcode);
```

The Shortcodes

```
[short_code_name type="cars"]
[short_code_name type="boats"]
[short_code_name type="cars" debug="1"]
```

As you can see, a single Shortcode can contain many "rules" before executing the code. As long as you capture the output, it really makes no difference what you do. You could connect to Mysql, run advanced math, call other functions and classes, include other files, and much more - all within one Shortcode.

This does not mean you should stack everything into a single Shortcode, but it does mean it's possible!

What one of my Shortcode functions looks like

This is one of my actual Shortcode functions. I pass in several Attributes in, however I only call one or few at a time, each of which manipulate the results of the included api.php file.

The *show* variable in the shortcode_atts array is set to *full* (show=full), this means the default value of show to full, thus if nothing is passed in it's value is full, otherwise its value is whatever is passed in from the Shortcode.

```
function apiData_func( $atts ) {
    extract( shortcode_atts( array('type' => "", 'show' => 'full', 'letter' => "", 'site' => "", 'start' => "", 'search' =>
", 'speed' => "", 'filter' => 'filter', ), $atts ) );
    ob_start();
    include('api.php');
    $output_string = ob_get_contents();
    ob_end_clean();
    return $output_string;
}
add_shortcode( 'api_data', 'apiData_func' );
```

The Shortcode

```
[api_data type="golf" letter="a" start="1"]
```

This Shortcode would display golf products, starting with the letter A on page 1, and would 'show' the 'full' information template.

Enable Shortcodes in Sidebar Widgets

Add the line below (as is) to your themes *function.php* file to turn on Shortcodes within Widgets. Now the same [ShortCodes] used within Posts and Pages can be used within Sidebar Widgets.

```
add_filter('widget_text', 'do_shortcode', 11);
```

What is the 11?

That is the Priority number, this tells Wordpress to load [do_shortcode](#) after the Wordpress content has started to load. In some unique cases you may need the Shortcode to load before Wordpress, like when modifying headers, then the Priority number would be 9 (I believe). Other times you may have nested the Shortcodes, and need them to execute in an order. Typically, it stays 11 or gets excluded all together.

Shortcodes are the ultimate Wordpress Shortcuts and when it comes to API's - Shortcodes are a godsend!

[Follow techNerdia on Google+](#) | [Follow Chris on Google+](#)
[Subscribe to techNerdia's RSS Feed](#)

Copyright 2012 © [techNerdia](#) - All Rights Reserved
<http://techNerdia.com/>