How To Use This Guide

Code examples are given in green and are very basic. Their attributes and inputs are shown as the attribute or input name. For example,

echo svg\_icon('icon', size, 'color');

“echo” is a PHP command that prints something on the page.

svg\_icon() is the function being called

'icon', size, and color are the attributes/inputs for the svg\_icon() function.

In this example the word 'icon' is the name of the input, and NOT a valid input. A proper input for 'icon' would be 'facebook'.

Inputs with single quotes around them should have single quotes around them when you call your function. Inputs without single quotes should not have quotes around them. To make it easy, strings ALWAYS have quotes. Boolians (true or false) NEVER have quotes. Numbers can have quotes, but it's optional. I leave them off because the color of a number and a string are different in most code editors, and you can more quickly tell if you have an error this way.

**Attributes**, **Inputs**, and **replacement** keys that you can and should use are in **bold**.

When the attributes are in an array, an example with all possible attributes will be in orange, like this:

$news\_options = array(  
 'number\_of\_news\_items' => '4',  
 'desc\_text\_limit' => '120',  
 'desc\_text\_hard\_break' => false,  
 'desc\_strip\_tags' => true,  
 'read\_more\_link' => false,  
 'title\_as\_link' => false,  
 'open\_links\_in\_new\_tab' => false,  
 'format\_month' => 'F',  
 'format\_day' => 'l',  
 'format\_year' => 'Y',  
 'format\_full\_date' => 'M j Y',  
 'template\_file' => 'name-news.php',  
 'news\_categories' => array()  
);

In the listing of attributes and inputs, the item in brackets will tell you what each attribute is expecting.

[STRING] – a series of letters and or numbers. A string must start with a letter. Strings are always surrounded in 'single quotes'

[TRUE OR FALSE] – Just the word true or the word false. NEVER surrounded by quotes.

[ARRAY] – a PHP array. Arrays are special variables that hold multiple sets of information, and are written like this:

$my\_array = array(

'key1' => 'value1',

'key2' => 'value2',

'key3' => 'value3'

);

When trying to echo value2 from the above array, one would write

echo $my\_array['key2'];

Many functions require a template file. Template files are used with the replacement keys to describe one instance of an element. For example, if I wanted every news element to have the title, followed by the date, followed by a description, and completing with a link inviting the visitor to read more, I would write the template file like this:

<div class=”news-item”>

<h3 class=”news-title”>**##TITLE##**</h3>

<span class=”news-date”>Posted on **##DATE##**</span>

<div class=”news-description”>

**##DESCRIPTION##**

</div>

</div>

Classes are up to the framework builder, but please make your classes descriptive!! A class of “nd” does not tell me if this is for a news description, a news date, or a nose digger. Long descriptive class names don't cost any more than short confusing ones.

Some functions need to be echo'd on the page, and some do not. Functions that need to be echo'd will say so. Every PHP function must be between PHP tags. When echoing out a single function in a sea of regular HTML, one would write the function like this:

<div>

regular html....

<? echo svg\_icon('facebook', 23, 'black'); ?>

</div>

In this instance, one can replace the word “echo” for its shortcut, =, like this:

<div>

regular html...

<?=svg\_icon('facebook',23,'black');?>

</div>

One last thing to keep in mind- lines of PHP are always ended with a ; (semi-colon). Before telling Jake you have an error, make sure there is a ; at the end of each line of PHP.

PHP Functions

1. Event Feed
   * Purpose: Display a feed of events on a client's page
   * Usage:
     + Step one is to set your Event Options. This is an array that can be set with your options at any point before the function call. An example of this array with all options listed is in the default config file.
     + Step two is to create your template file. This will be the simple HTML file with the event replacement keys. This template file should describe exactly what one single calendar event should look like when displayed on the page. Use the replacement (listed later in this set of instructions) keys to put in the correct information.
     + Step three is to call the function on your page and pass it your options. The function call looks like this:

event\_feed($event\_options);

($event\_options is the name you gave to your event options array)

* + - Note – The event feed has lots of options and the replacement keys reflect every piece of information the client could possibly give. I usually only use ##TITLE##, ##DATE##, and ##DESCRIPTION## in most of my feeds. Try only setting the attributes that you need.
  + Attributes & Inputs (Event Options)
    - **number\_of\_events** – [NUMBER] the number of events to display on a page
    - **desc\_text\_limit** – [TRUE OR FALSE]The amount of text (in number of characters) to display for each event description. If the text is cut off, the script will add an ellipses (...)
    - **desc\_text\_hard\_break** – [TRUE OR FALSE] If true, text will be cut off exactly after the character set in the desc\_text\_limit. If false, it will go to the end of the word. False usually looks better.
    - **desc\_strip\_tags** – [TRUE OR FALSE] If set to true, this will remove any HTML from the description before displaying it. Very useful if someone has the ability to add links or images.
    - **read\_more\_link** – [STRING OR FALSE] Use this to set the display text for the “read more/more info” link. Set it to false to not display this link by default.
    - **title\_as\_link** – [TRUE OR FALSE] If true, the title will be a link to the full info page. If false, the title will be plain text.
    - **open\_links\_in\_new\_tab** – [TRUE OR FALSE] If true, title and “more info” links will open in a new tab/window. If false, they'll open in the active tab/window.
    - **format\_month** [STRING] Set the display of the month (Oct, October, 10, etc). For more information on PHP dates, please see this page: <http://php.net/manual/en/function.date.php>
    - **format\_day** – [STRING] See format\_month
    - **format\_year** – [STRING] See format\_month
    - **format\_full\_date** – [STRING] See format\_month
    - **template\_file** – [STRING] The name and path of your event template file, such as “my\_template.php” or “templates/my\_template.php”
    - **cal\_type** – [ARRAY] Leave this blank to show all events, or set an array of the calendar types (found in calendar manager) that you would like to include.
    - **highlight\_class** – [STRING] Clients have a “highlight” function when creating events. Set this to the CSS class you would like applied to highlighted items.
  + Attribute Example:

$event\_options = array(  
 'number\_of\_events' => '3',  
 'desc\_text\_limit' => '80',   
 'desc\_text\_hard\_break' => true,  
 'desc\_strip\_tags' => true,  
 'read\_more\_link' => false,  
 'title\_as\_link' => true,  
 'open\_links\_in\_new\_tab' => false,  
 'format\_month' => 'F',  
 'format\_day' => 'l',  
 'format\_year' => 'Y',  
 'format\_full\_date' => 'M j Y',  
 'template\_file' => 'name-events.php',  
 'cal\_type' => array(),  
 'highlight\_class' => 'ev-highlight'  
);

* + Replacement Keys – Use these when creating your template file
    - **##TITLE##**
    - **##DESCRIPTION##** - the primary way people talk about their event
    - **##DETAILS##** - Secondary, many people do not use
    - **##MORE\_LINK##** - link to full details (used by read\_more\_link from attributes)
    - **##URL##** - A raw link to the info page if you want to use it in a fun way
    - **##LINK\_1\_TITLE##** - Title of link 1, set in Calendar Admin
    - **##LINK\_1\_URL##** - URL of link 1
    - **##LINK\_2\_TITLE##**
    - **##LINK\_2\_URL##**
    - **##HIGHLIGHT##** - adds the class you set in highlight\_class
    - **##START\_TIME##**
    - **##END\_TIME##**
    - **##DAY##** - Displays the start day of the event
    - **##MONTH##**
    - **##YEAR##**
    - **##DATE##** - Displays the full start date of the event
    - **##CONTACT\_NAME##**
    - **##CONTACT\_EMAIL##** - Not a link! Just the email address
    - **##CONTACT\_PHONE##** - Not a link! Just the Phone Number
    - **##LOCATION\_NAME##** - Name of Event Location
    - **##ADDRESS\_1##** - Address line 1
    - **##ADDRESS\_2##** - Address line 2 – only displays if it has a value
    - **##CITY##**
    - **##STATE##**
    - **##ZIP##**

1. Input Shortener
   * Purpose: Take user submitted text and shorten it to a desired length
   * Usage:
     + Call this function in the location where you would like to display an exact amount of text submitted by the client.
     + Call the function like this:

input\_shortener('page', end, start);

* + Attributes and Inputs
    - **page** [STRING – REQUIRED] This is client submitted page you would like to display. Use the HTM version of each page.
    - **end** [NUMBER – REQUIRED] – The number of characters to display.
    - **start** [NUMBER – OPTIONAL] – Defaults to the first character. Can be set to a different starting character.

1. Log In Form - Simple
   * Purpose – Display a log in form
   * Usage:
     + Echo this function wherever you would like to display a login form.
     + Step One: set any options you would like to change from their default behavior in an array.
     + If displaying a single log in form, call the function like this:

echo log\_in\_form($log\_in\_options\_array);

* + - If displaying multiple log in forms, set a variable and echo the variable for each instance of the form, like so:

$my\_log\_in\_form = log\_in\_form($log\_in\_options\_array);

then call the variable like this:

echo $my\_log\_in\_form;

* + - Note: The log in form has many attributes that you can change, but has a set of defaults. Try calling the log in form without options to see how it looks by default, and only changing the options you need.
  + Attribute Example

$log\_in\_options\_array = array(

'get' => form,

'layout' => vertical,

'user\_placeholder' => 'Username',

'user\_label' => 'Username',

'user\_class' => 'username-css-class',

'pass\_placeholder' => 'Password',

'pass\_label' => 'Password',

'pass\_class' => 'password-css-class',

'forgot\_pass' => true,

'forgot\_pass\_label' => 'Forgot Password?',

'forgot\_pass\_position' => 'left',

'forgot\_pass\_class' => 'forgot-pass-css-class',

'submit\_value' => 'Submit',

'submit\_class' => 'submit-button-css-class'

);

* + Attributes and Inputs
    - **get** – [STRING] Either “Form” or “Pieces” - pieces will be explained next
    - **layout** – [STRING] either “vertical” or nothing. If set to vertical, additional divs will be added to stack the form, otherwise the form returns on one line
    - **user\_placeholder** [STRING OR BLANK] – The text entered here will appear inside the username input box.
    - **user\_label** [STRING OR BLANK] – The text entered here will appear to the left of the username input box.
    - **user\_class** [STRING OR BLANK] – Applies a specified class to the username input box
    - **pass\_placeholder** [STRING OR BLANK] – The text entered here will appear inside the password input box.
    - **pass\_label** [STRING OR BLANK] – The text entered here will appear to the right of the password box.
    - **pass\_class** [STRING OR BLANK] – Applies a specified class to the password input box
    - **forgot\_pass** [TRUE OR FALSE] – Displays the “Forgot Password?” link for password recovery.
    - **forgot\_pass\_label** [STRING] – Sets the display text of the forgot password link.
    - **forgot\_pass\_position** [left OR right] – Puts the forgot password link to the left or right of the submit button
    - **forgot\_pass\_class** [STRING] – Adds the class you specify here to the forgot password link
    - **submit\_value** [STRING] – sets the display text for the submit button
    - **submit\_class** [STRING] – Adds the class you specify here to the submit button

1. Log In Form – Pieces
   * Purpose – Use the log in form function to get the pieces of the log in form to use in a custom format. If you're trying something wild for a log in form, you may not want the form in the order that is normally returned. Set the 'get' option for the function to 'pieces'
   * usage:
     + Step one – set the 'get' option to 'pieces'
     + step two – call set a variable to call the log in form to create an array, like this:

$my\_var = log\_in\_form($options);

* + - Step Three – call the individual pieces of the log in form as an array, using the echo command. The useable pieces from the example above are these:
      * **$my\_var['form\_start']** – Starts the form
      * **$my\_var['input\_user']** – Displays the username input
      * **$my\_var['input\_pass']** – Displays the password input
      * **$my\_var['input\_submit']** – Displays the submit button
      * **$my\_var['forgot\_pass']** – Displays the Forgot Password link
      * **$my\_var['form\_end']** – Ends the form
    - Note: With the exception of the $my\_var['forgot\_pass'] piece which is not part of the form submission, every other piece must be called between $my\_var['form\_start'] and $my\_var ['form\_end']

1. Menus
   * Purpose: Display one of the 4 menu types on the page.
   * Menu Type 1: Quick Menu – quick\_menu();
     + Purpose: Main navigation
     + Usage
       - Call the Quick Menu where you would like to display the main page navigation, like this:

quick\_menu('before', 'after', 'custom caret');

* + - Attributes and Inputs
      * **before** [OPTIONAL STRING] – Add <li> items directly to the start of the menu. You must use the <li> and </li> tags for each element you wish to add before the start of the normal menu.
      * **after** [OPTIONAL STRING] – add <li> items directly to the end of the menu. You must use the <li> and </li> tags for each element you wish to add before the end of the normal menu.
      * **Custom Caret** [OPTIONAL STRING] – If you don't like the default down arrow, you have the option to make your own.
  + Menu Type 2: Mobile Menu – mobile\_menu();
    - Purpose: Add a default mobile menu for smaller media sizes
    - Usage
      * Part one – call the function. The function accepts an array of options detailed in the next section. Call the mobile\_menu function in either the header or footer like so:

mobile\_menu($options\_array);

* + - * Part two – add the triggers. The trigger is the area where someone clicks to initiate the mobile menu. It is just a CSS class that can be added to just about any element like this:

<div class=”mobileMenuTrigger”>[whatever button]</div>

or

<a class=”mobileMenuTrigger”>[button]</a>

* + - Attributes and Inputs
      * **before** [OPTIONAL STRING] – Add <LI> items or <DIV> items before the start of the menu. Adding a div with a logo is a great way to brand this menu. Adding the search\_box() is another great option.
      * **after** [OPTIONAL STRING] – Add <LI> items or <DIV> items after the last item in the menu.
      * **IMPORTANT NOTE – iPhone and iPad must have a trigger inside the menu. Please use the before or after option to add a trigger element.**
  + Menu Type 3: Sub Menu – sub\_menu();
    - Purpose: display a sub menu of the current page up to its top level link. This is primarily used on sub pages. If the top level link is “About Us”, sub\_menu() will display a menu with “About Us” at the top, and any sub-pages, including the one you're on, that are associated with the About Us top-level navigation item
    - Usage
      * Call the sub\_menu() where you would like to display a sub-menu like so:

sub\_menu();

* + - Attributes and Inputs – NONE (this is an easy one).
  + Menu Type 4: Breadcrumb Menu – breadcrumb\_menu();
    - Purpose: Display the breadcrumb menu leading back to home.
    - Usage
      * Call the breadcrumb menu wherever you would like to display said menu like this:

breadcrumb\_menu(' separator ', 'use index');

* + - Attributes and Inputs
      * **Separator** [STRING] – Defaults to ' > '. Please note the spaces! If you want spaces around your separator, you must add them!
      * **Use index** [STRING or FALSE] – If False, the breadcrumb menu will NOT start with a link to the front page. If given a string like, “home”, the menu will start with a link to the index.php page and use the text you provide in this input.

1. News Feed
   * Purpose: Display a feed of news items
   * Usage
     + Step One – Create an HTML template for one news item using the replacement keys listed later under attributes
     + Step Two – Set the options for your news feed before you call the function. The options will be listed under attributes
     + Step Three – Call the function like this:

<? news\_feed($news\_options); ?>

* + Attributes and Inputs
    - News Options – set these before you call news\_feed();
      * **number\_of\_news\_items** [NUMBER or FALSE] – If a number is provided, that many of news items will show. If FALSE is passed, every news item will show.
      * **desc\_text\_limit** [NUMBER or FALSE] – If a number is provided, that many characters will show in the description. If FALSE is passed, the entire description will show.
      * **desc\_text\_hard\_break** [TRUE OR FALSE] – If true, the description will cut off after the character given in desc\_text\_limit. If false, the description will cut off after the full word located at the desc\_text\_limit
      * **read\_more\_link** [STRING OR FALSE] – If false, the read\_more\_link will not show by default. If you enter a string like 'Read More', the link text will default to 'Read More'
      * **title\_as\_link** [TRUE or FALSE] – Sets the news title to display as a link to the full version of the news article
      * **open\_links\_in\_a\_new\_tab** [TRUE OR FALSE] – sets the links to the full article to open in a new tab or not.
      * **format\_month** [STRING] – see <http://php.net/manual/en/function.date.php> for details on formatting dates
      * **format\_day** [STRING]
      * **format\_year** [STRING]
      * **format\_full\_date** [STRING]
      * **template\_file** [STRING] – location and name of the template file to use.
      * **news\_categories** [ARRAY] – Usually left empty to display the most recent news, you can set this to an array of news categories if you'd like the feed to only include certain categories.
    - Attributes example:

$news\_options = array(  
 'number\_of\_news\_items' => '4',  
 'desc\_text\_limit' => '120',  
 'desc\_text\_hard\_break' => false,  
 'desc\_strip\_tags' => true,  
 'read\_more\_link' => false,  
 'title\_as\_link' => false,  
 'open\_links\_in\_new\_tab' => false,  
 'format\_month' => 'F',  
 'format\_day' => 'l',  
 'format\_year' => 'Y',  
 'format\_full\_date' => 'M j Y',  
 'template\_file' => 'name-news.php',  
 'news\_categories' => array()  
);

* + - News Template Replacement Keys
      * **##DATE##** - Displays the full date formatted in the options
      * **##DAY##** - Displays the day formatted in the options
      * **##MONTH##** - Displays the month formatted in the options
      * **##YEAR##** - Displays the year formatted in the options
      * **##TITLE##** - Displays the News Item Title
      * **##DESCRIPTION##** - Displays the description
      * **##URL##** - Returns a URL to the full version of the article
      * **##MORE\_LINK##** - Displays a link to the full version of the article with the text from the read\_more option
      * **##IMAGE##** - Returns an HTML image tag with the news item's image if one exists.

1. RSS News Feed
   * Purpose: Grab an RSS feed from another location and post the items on a page.
   * Usage
     + Step One – Create an HTML template for one news item using the replacement keys listed later under attributes
     + Step Two – Set the options for your RSS news feed before you call the function. The options will be listed under attributes
     + Step Three – Call the function like this:

rss\_news\_feed($news\_options);

* + Attributes and Inputs
    - News Options – set these before you call news\_feed();
      * **strip\_tags** [TRUE or FALSE] – Defaults to true. If set to false, the description will keep all of its tags.
      * **strip\_tags\_exceptions** [STRING] – If strip\_tags is set to true, you can leave some tags in. If you wanted to keep <br> tags, since some people write them differently, you would write the exceptions like this:

'strip\_tags\_exceptions' => '<br><br/><BR><BR/>'

* + - * **feed\_url** [STRING] – The full URL to the XML RSS feed.
      * **number\_of\_news\_items** [NUMBER or FALSE] – If a number is provided, that many of news items will show. If FALSE is passed, every news item will show.
      * **desc\_text\_limit** [NUMBER or FALSE] – If a number is provided, that many characters will show in the description. If FALSE is passed, the entire description will show.
      * **desc\_text\_hard\_break** [TRUE OR FALSE] – If true, the description will cut off after the character given in desc\_text\_limit. If false, the description will cut off after the full word located at the desc\_text\_limit
      * **read\_more\_link** [STRING OR FALSE] – If false, the read\_more\_link will not show. If you enter a string like 'Read More', the link text will be 'Read More'
      * **title\_as\_link** [TRUE or FALSE] – Sets the news title to display as a link to the full version of the news article
      * **open\_links\_in\_a\_new\_tab** [TRUE OR FALSE] – sets the links to the full article to open in a new tab or not.
      * **format\_month** [STRING] – see <http://php.net/manual/en/function.date.php> for details on formatting dates
      * **format\_day** [STRING]
      * **format\_year** [STRING]
      * **format\_full\_date** [STRING]
      * **template\_file** [STRING] – location and name of the template file to use.
    - RSS News Attributes Example

$rss\_options = array(  
 'strip\_tags' => false,  
 'feed\_url' => 'rss feed address.xml',  
 'number\_of\_news\_items' => '2',  
 'desc\_text\_limit' => false,  
 'desc\_text\_hard\_break' => false,  
 'read\_more\_link' => false,  
 'title\_as\_link' => true,  
 'open\_links\_in\_a\_new\_tab' => 'Password&hellip;',  
 'format\_month' => 'F',  
 'format\_day' => 'l',  
 'format\_year' => 'Y',  
 'format\_full\_date' => 'M j Y',  
 'template\_file' => 'rss\_template.php'  
);

* + - RSS News Template Replacement Keys
      * **##DATE##** - Displays the full date formatted in the options
      * **##DAY##** - Displays the day formatted in the options
      * **##MONTH##** - Displays the month formatted in the options
      * **##YEAR##** - Displays the year formatted in the options
      * **##TITLE##** - Displays the News Item Title
      * **##DESCRIPTION##** - Displays the description
      * **##URL##** - Returns a URL to the full version of the article
      * **##MORE\_LINK##** - Displays a link to the full version of the article with the text from the read\_more option

1. Search Box
   * Purpose: Display a google CSE search box on the page
   * Usage
     + Display the search box by calling the function like so:

search\_box($options\_array);

* + Attributes and Inputs
    - **placeholder** [OPTIONAL | STRING] – Set this to what you would like the search box to say.
    - **bootstrap-button** [OPTIONAL | left OR right OR FALSE] – If set to left or right, this will add the bootstrap classes and a wrapper around the search box. Left shows the submit button on the left, right shows the submit button on the right, and FALSE removes the submit button (forcing the user to press enter/return)
    - **submit\_button** [OPTIONAL STRING] – The text of the submit button if you'd like something other than the default, “Submit”
  + Attribute example:

$search\_box\_options = array(

'placeholder' => 'Search',

'bootstrap-button' => 'left',

'submit\_button' => '<div class=”submit-button”>Submit!</div>'

);

1. Slideshows
   * Purpose: Display one of the two slideshows available
   * Usage
     + Quick Slideshow Purpose: display a regular slideshow. Set the left and right button styles with CSS
     + Quick Slideshow Usage – Call the function like this:

echo quick\_slideshow('slideshow', speed);

* + - Quick Slideshow Attributes and Inputs
      * **slideshow** [STRING] – The slideshow number found on the front page of the slideshow manager. This will always start with a capital “FD” and end with the number of the slideshow, such as 'FD1' or 'FD10'
      * **speed** [NUMBER] – the number of milliseconds you would like each slide to display for. For example, a five second duration would be set to 5000.
    - Feature Box Slideshow Purpose: When feature boxes have a photo, a title, and a small amount of text, use the slideshow manager to make it so clients can't add too much or too little
    - Feature Box Slideshow Usage
      * Step One – Create the template file and upload it. Template keys will be listed later under attributes
      * Step Two – Call the featurebox slideshow function like this:

<? echo feature\_box\_slideshow('Slideshow', number of boxes, 'template file'); ?>

* + - Feature Box Slideshow Attributes and Inputs
      * **Slideshow** [STRING] - The slideshow number found on the front page of the slideshow manager. This will always start with a capital “FD” and end with the number of the slideshow, such as 'FD1' or 'FD10'
      * **number of boxes** [NUMBER] – How many feature boxes to display. Clients can enter any number of slides, and this limits all of them displaying on the page.
      * **Template file** [STRING] – This is the location and name of the template file you create, such as 'fb\_template.php' or 'templates/fb\_template.php'
      * Template Replacement Keys
        + **##TITLE##** - pulls information from the slideshow Caption input
        + **##TEXT##** - pulls information from the slideshow Alt Text input
        + **##LINK##** - pulls information from the slideshow URL input
        + **##IMAGE##** - outputs the images source, to be used like:

<img src=”##IMAGE##” />

1. SVG Icons
   * Purpose: Display SVG icons of any size or color
   * Usage – as php function
     + To display an SVG image, call the function like this:

echo svg\_icon('icon', size, color);

* + Attributes and Inputs as a PHP function
    - **icon** [STRING] – The name of the SVG Icon. Case sensitive!
    - **size** [NUMBER] – The icon's height in number of pixels. Icons default to 25 pixels tall and adjust their width automatically, but since they're SVG icons, make them as large or small as you need!
    - **Color** [HTML/CSS COLOR STRING] – The color can be given in HEX, RGB, or RGBA. When using HEX, don't forget the #!!
  + Usage – as an image
    - If you're using the SVG icon as a background image in CSS, or you just have a thing for <img> tags, call the icon like this:

<img src=”images/svg.php?i=icon&s=size&c=color” />

* + Attributes and Inputs as an Image:
    - **i=icon** [STRING] – replace the word “icon” with the icon you'd like to call
    - **s=size** [NUMBER] – replace the word “size” with the desired height of the icon
    - **c=color** [HTML COLOR STRING] – calling the image this way limits us to the HEX color, without the #.

JavaScript Functions

1. Equal Height
   * Purpose: Make any block-level elements the same height. Very useful for sidebars, and often necessary for vertical centering without tables.
   * Usage
     + Any block-level HTML elements that you would like to display at the same height will need the equal-height class, a data-group, and a data-to, like this:

<div class=”equal-height” data-group=”groupName” data-to=”770”>

…

</div>

* + - Each element that needs equal-height applied will need each of the attributes
  + Attributes and Inputs
    - **equal-height** – Apply to the element's class.
    - **data-group=”group”** - Elements that should be the same height need to be in the same group.
    - **data-to=”770”** - In this case, the equal-height elements will return to their default height at a screen width of 770 pixels. This is usually set to the desired bootstrap break point.

1. Format Phone Link
   * Purpose: mobile devices and PC's with telephony software installed can take advantage of the new href designation of tel:. Setting a link to

<a href=”tel:5175559876”>Call Us</a>

will open the telephony program of the device and start calling 517.555.9867

The Phone Link script was written so that clients can't mess this up

* + Usage – Class based
    - Step One – In the content manager, create a file for the phone number and give it a title like “Footer Phone Number,” which creates footer\_phone\_number.htm on the client's server
    - Step Two – Include a link, in this example, I would be adding the link in the footer like this:

<a href=”” class=”phone-link”>

<? readfile('footer\_phone\_number.htm');?>

</a>

* + - Note: Clients can enter any characters along with the phone number, such as (800) 555-5555, or 800.555.5555. The script will remove these extra characters when adding the link to the anchor tag, but display them for the visual output.

1. Item Slider
   * Purpose: Add a control set to sibling elements. ISControl is the trigger that will perform the slide-up or slide-down on ISContent.
   * Usage – Class based
     + Step One – Create two sibling elements. It's a good idea to put these in a wrapper so they don't conflict with any of your other Item Slider elements.
     + Step Two – Add the necessary ISControl and ISContent classes to your block level elements, like so:

<div class=”wrapper”>

<h2 class=”ISControl”>Events</h2>

<div class=”ISContent”>

…

</div>

</div>

* + - **ISControl** – class, added to element that should be clicked
    - **ISContent** – class, added to element to be shown or hidden when clicking ISControl element.

1. Link Converter
   * Purpose: Convert a client-submitted page of links into an unlimited list and list options
   * Usage: class link-convert
     + Step One – In the content manager, create a file and call it something like, Quick Links.
     + Step Two – Where you want to display the list of links, apply the class link-convert, and call the page, like this:

<div class=”link-convert”>

<? readfile('quick\_links.htm'); ?>

</div>

* + - Step Three – Style the <ul> and <li>'s in your CSS

1. Nav Padder
   * Purpose: give an equal distance between top-level navigation items
   * Usage: automatic.
     + Nav Padder is turned on or off in your config.php file. It requires no other input from you.
2. User Image Height Width Fix
   * Purpose: When including a content file on a page where clients have the ability to place images, it's sometimes important that those images do not flow over the bounding boxes of their content. CKEditor from the content manager hard-codes height and width attributes to each image it writes. This script removes those hard-coded heights and widths, allowing you to make the images responsive.
   * Usage: class – user
     + Apply the class “user” to any content wrapper where a client may submit a photo that could break bounding boxes, like this:

<div class=”user”>

<? readfile('user\_submitted\_content.htm'); ?>

</div>

1. Vertical Center (vcenter)
   * Purpose: Tables allow you to vertically center an item, but tables are non-responsive. This script allows you to vertically center one element (target element) inside another element (parent element).
   * Usage: class vcenter
     + Apply the class vcenter to any target element that should be centered inside its parent element
     + Note: the parent element MUST have a height set.
     + Note: This class is often used on elements inside of parent elements that have the equal-height class/script applied to them

<div height=”300px”>

<div height=”100px” class=”vcenter”> … </div>

</div>