INSTALLING AND CONFIGURING THE **CITIZEN PRESS THEME** FOR WORDPRESS

USER'S GUIDE | VERSION 1.0

CONTENTS

1 INTRODUCTION

Welcome! ☺ This guide was written to help you get started with using the Citizen Press theme for WordPress. Citizen Press is more than just a theme – it's a framework! If you are a beginning to intermediate web user, you'll find that the Citizen Press options panel offers you many choices that help you build a unique site. If you're an advanced user, you can bend the theme to your will even more by taking advantage of the many built-in action and filter hooks. Without further ado, let's get started, shall we?

2 BASIC REQUIREMENTS

First things first! Let's go over the requirements and assumptions that I'll be making throughout this guide:

- You have **WordPress 3.0 or higher** installed and running on your own server (it won't work with any version prior to 3.0). Citizen Press only works with self-hosted WordPress (http://www.wordpress.org), NOT WordPress.com.
- If you plan to use the theme options panel, your browser has JavaScript turned on.
- You are familiar with the WordPress interface and lingo. If you aren't, the WordPress Codex (codex.wordpress.org) is a good place to start.
- You at least know *what* FTP, HTML, CSS, JavaScript and PHP are. The more proficient you are in those areas, the better. I provide links to helpful web design resources in Section 11 of this guide! ☺
- You understand (or are willing to understand) the relationship between Parent/Child Themes in WordPress (codex.wordpress.org/Child_Themes).

2.1 Make sure Citizen Press is indeed the right choice!

Citizen Press was designed for newspaper and magazine websites — sites with a lot of content. It *can* be scaled down to a blog format, but keep in mind that it is primarily a theme for newspapers/magazines.

If you're looking for a theme that "works-out-of-the-box-in-five-minutes", with minimal configuration, Citizen Press is not for you. Of course, pre-made Child Themes can provide ready-made designs, but even if you use a Child Theme, you will still need to configure some settings. Citizen Press puts *you* in charge of deciding which elements should go where on your front page and in the sidebars on interior pages. You'll need to spend some time figuring out how you want to lay out your content and be willing to experiment before you get the right look. Citizen Press is designed to be flexible enough to serve each paper's unique needs – it's not meant to be a canned, one-size-fits-all solution.

Like most theme frameworks, Citizen Press comes with a set of its own conventions and lingo (it's simple, I promise). Therefore, expect a slight learning curve in the beginning. Finally, Citizen Press uses a CSS grid system to control the layout structure and column widths. CSS grids provide a solid platform for anyone to create multi-column websites with ease. Grids also help keep elements aligned nicely on the page and aid in a consistent, readable feel throughout. I'll discuss grids in later sections of this guide. If you don't like the idea of working with CSS grids, Citizen Press may not be for you.

2.2 Citizen Press known code issues

In addition to the considerations listed above, Citizen Press contains a number of known issues regarding its coding. A disclaimer: I don't claim to be an expert in PHP. I've learned most of what I do know as I've gone along and have taken care to code it to the best of my ability using the best practices that I am aware of. I'm releasing Citizen Press because I'm very confident in what it can do and in its potential. It's also my way of giving back to the WordPress community.

However, that doesn't mean the code doesn't have its share of issues; things that I haven't been able to fix as of the time this guide was written (but hope to fix in the future). I've listed all of the known issues in the online documentation at the Citizen Press website (http://pixelplanethemes.com/). I highly recommend that you read through them. That way, if any of them are showstoppers for you, you'll be aware *before* you invest your time and energy.

I brought all of this up so that you know what to expect before you start. So, let's review:

Citizen Press was made for those who…

- Aren't looking for a one-size-fits-all, plug-and-play framework where all the sites based on it look the same
- Don't mind spending time configuring options to get that unique look you want!
- Are hoping to create a news/magazine style site rather than a blog or corporate site
- Want a versatile, powerful newspaper theme framework that offers a huge amount of flexibility, for non-coders and coders alike.
- Don't mind working with CSS grids.
- Don't mind working with a theme framework and Child Themes.
- Know basic HTML, CSS and PHP (or are willing to learn/have someone else do this part) to make customizations beyond the options panels (unless you're willing to have someone else do this part for you).
- Have JavaScript turned on (if they plan to use the theme options panel)
- Want a theme where nearly *everything* can be changed without having to touch the parent theme's code. With Citizen Press (and with most theme frameworks), if it can't be changed in the options panel, you can bet that it can be changed through CSS, action hooks, filter hooks and/or pluggable functions. And in the event that you do need to change an actual template

file, you can just copy said file over to your Child Theme, and the parent theme's code is *still* unchanged.

2.3 If you're not a coder/developer…

First off, don't be alarmed by the number of options on the Citizen Press options page. You certainly don't need to use them all. Just take your time configuring the most important settings (this guide AND the "lightbulb links" in the options panel will help you along the way). The options that you don't use will be there should you need them later. There are also resources on the Pixel Plane Themes website (tutorials, references and forums) to help you.

With Citizen Press, you can set column widths and arrange newsboxes and sidebars without touching code. With the Colorific Child Theme, you get a nice set of color pickers and font face/size size options that let you make basic design changes to many elements of the theme without knowing CSS!

Keep in mind that while you *do* have a large amount of flexibility in the options panels and can create a great looking site with those alone, there are still limits. To take full advantage of the power offered by Citizen Press, you will need to spend some time learning basic HTML and CSS so that you can edit the stylesheet beyond the limits of Colorific (see Section 11 for some helpful web design resources). And if you pick up basic PHP, you will be able to work with the many action and filter hooks in the theme so that you can drop your own custom content into the mix.

If you'd rather not dabble in code, then using a pre-made Child Theme or hiring a web designer to build one for you would be possible routes.

2.4 If you're developer…

Citizen Press is jam-packed with action and filter hooks. Using these hooks, you can create Child Themes that rearrange elements on the page and add your

own custom content wherever you wish. Be sure to read the "Hooks and Pluggable Functions" section later on in this guide for more information.

3 QUICK START GUIDE

This section outlines the steps needed to install and configure a typical setup of the Citizen Press theme. Before we start, however, there are two main concepts that you should be familiar with, as they will pop-up often in this guide AND on the Citizen Press options panels – *templatebits* and the *grid system*. Let's go over them quickly.

3.1 Templatebits

Templatebits are the building blocks of the Citizen Press theme. They are basically everything you see on the front page: the menu, the sidebars, the newsboxes, you name it. On single posts, archives and pages, templatebits are everything except for the post/page/archive content area. I call them "templatebits" because they are, quite literally, bits of a template. Instead of hard-wiring everything into the theme files, I cut the theme up into bits (small individual files) and we're left with templatebits.

Some templatebits are "built-in", which means they have pre-defined default locations that you can't change in the options panel (however, some have options that let you modify their content or hide them altogether, but to move their actual location, you'll need to use action hooks).

The rest of the templatebits are "modular", meaning it's up to you to decide where to place them. You'll place them in designated regions on the front page, single posts, single pages and archive pages. Let me cut to the quick and identify the templatebits and tell you where they're located.

Where to find Templatebits: All templatebits are located in *citizenpress/library/templatebits*. The modular templatebits are in the root of

this folder. The built-in templatebits are in the "built-in" folder. It's as simple as that!

Identifying the Templatebits:

- **Built-In** (Note: some of these are disabled by default and must be turned on in the options panel):
    - *auxiliarybar.php* – This two-column bar contains the "all sections" drop-down box, the current date, the login/register links and a menu. By default, it appears at the top of the site on all pages.
    - *breaking-news.php* – This is an eye-catching banner that you can turn on via the options panel whenever you have a huge breaking story that you want to announce. By default, it appears on the front page only, underneath the main menu.
    - *footer-menu.php* – Self-explanatory, the menu in the footer of the site, where it appears on all pages by default.
    - *main-menu.php* – The main navigation menu, appears under the Masthead on all pages by default.
    - *masthead.php* – A two-column block that contains the site logo on the left by default; and a space for widgets on the right. Appears on all pages by default, under the auxiliary bar and above the main menu.
    - *news-ticker.php* – A jQuery-controlled ticker that rotates headlines from a category you specify. By default, it appears on the front page only, under the main menu, and below the breaking news banner if the breaking news banner is enabled.
    - *section-index.php* – A group of three additional menus you can use. By default it appears just above the footer menu on all pages.
    - *sidebar-archives.php* – A sidebar that only appears on archive listings (category, tag, date, etc). Appears on the right column by

default; above all other modular templatebits you may place in that column.

- *sidebar-horizontal.php* – A horizontal sidebar that can hold up to six widgets. By default, it appears on the front page only; just above the section index (if the section index is enabled) and below all the news and sidebar sections.
- *sidebar-pages.php* and *sidebar-posts.php* – Same deal as sidebar-archives.php above, except these sidebars appear only on pages and posts, respectively.

- **Modular** (you decide where these should go within designated areas):
  - *searchform.php* – Search form (Not to be confused with the search widget).
  - *sidebar-1.php, sidebar-2.php, sidebar-3.php, sidebar-4.php* – Four standard sidebars.
  - *sidebar-tabbed.php* – This sidebar displays widgets as tabs.
  - *featured-tag-box.php* – Lets you highlight posts based on a tag that you specify in the theme options panel.
  - *sidebar-double.php* – This is a two-column sidebar.
  - *vnewsbox-1.php, vnewsbox-2.php, vnewsbox-3.php* – These are standard newsboxes which display news stories in a vertical format. You configure them in the options panel.
  - *hnewsbox-1.php, hnewsbox-2.php, hnewsbox-3.php* – These are standard newsboxes which display news stories in a two-column horizontal format. You configure these in the options panel as well.
  - *top-story.php* – This is a special newsbox which lets you give more prominence to "top stories".
  - *latest-news.php* – This newsbox displays the latest headlines and their posting times by default.

3.2 Grid System

The Citizen Press theme uses a CSS grid for its underlying structure. CSS grids are the web equivalent of the grid structure often used in print design. Just as they do for a printed publication, grids provide websites with a consistent, intuitive organizational structure. Equal margins between elements throughout can enhance the ease by which a user can read and navigate the content. HTML tables once served as a common stand-in for creating grid-like structure for web pages, but the separation of content from design has shifted the trend toward the use of CSS for all aspects of layout. CSS grids provide a set of CSS style rules that define "columns" with pre-determined widths, margins and padding.

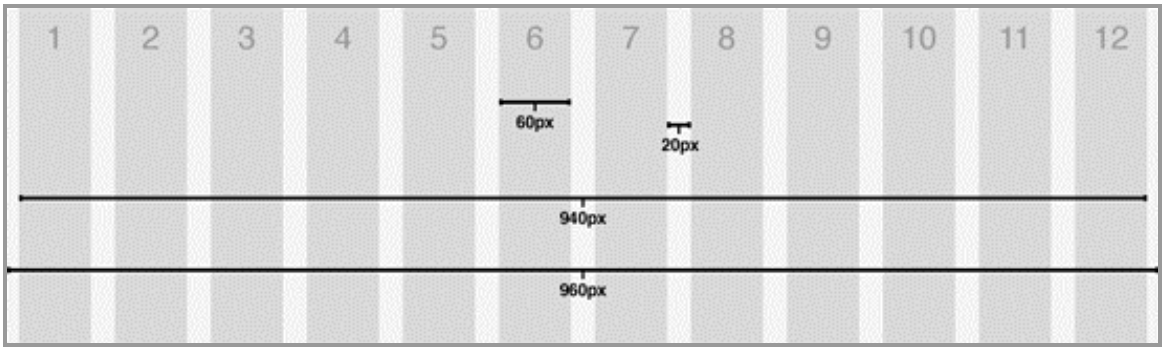
Simple grid systems can help streamline both the initial design and maintenance of a website. Once the website maintenance team learns how to use the grid, they can make rapid changes to the layout that are guaranteed to display properly without needing advanced HTML and CSS skills.

There are many CSS grid systems available on the web, ranging from simple to complex. For Citizen Press, I selected the 1KB CSS Grid by Tyler Tate. It is simple, lightweight and easy to use. Citizen Press offers the grid in four pre-built configurations as well as a "freeform" mode where you can create your own dynamic grid by specifying custom widths.

Pre-Built Configuration. By default, the grid is set to a pre-built configuration that consists of 12 possible columns of 60 pixels wide with a 20-pixel gutter in between each column (see picture on next page). Therefore, a column that is 1 grid unit wide contains 60 pixels of usable space not counting the gutter, and a column that is 2 grid units wide contains 140 pixels of usable space (2 60-pixel grid units plus 1 20-pixel gutter, or 60+60+20). A column that is 12 grid units wide contains 940 pixels of usable space (12 60-pixel columns plus 11 20-pixel gutters, or [60 x 12]+[20 x 11]). The overall site width for the default configuration of the grid is 960 pixels wide counting the gutters, and the content

width is 940 pixels.

1KB CSS Grid 12-60-960 (default) configuration.



The mark-up for the 1KB CSS grid system is simple and straightforward:

```
<div class="row">
   <div class="column grid_6">Content of this column</div>
   <div class="column grid_6">Content of this second column</div>
</div>
```

As you can see, each column must be nested inside a with the CSS class ".row". Furthermore, each column must contain two classes: ".column" (which designates this as a column) and ".grid_X" (where "X" is the number of grid units).

Important: The sum of the grid units for ALL columns in a single row must NOT exceed the maximum number of grid units. Throughout the Citizen Press theme, this number is referred to as “MAXGRID”. For the default grid configuration, MAXGRID is 12. Look at the code example posted above. There are two columns, each with a width of 6 grid units. 6+6 = 12. We could have created a row of three columns, each 4 grid units wide. Or we could have created a row with a single column, 12 grid units wide. But a row with three columns of grid units 8, 5 and 2 will produce quirky results. For best results, always make sure the sum of all grid units in a single row is less than or equal to MAXGRID.

The CSS for the default grid-related styles (.row, .column and .grid_X) can be found in *citizenpress/library/options_page/style-grid.php*. It is highly recommended that you do not alter any aspects of the grid styles.

The grid mark-up has been built in to the Citizen Press theme, so you don't need to write any of it yourself. If you're using a pre-built grid configuration, the only grid-related things you'll be asked to do in the Citizen Press options page are:

1. Picking a grid configuration. Each grid configuration contains a site width and maximum number of grid units (MAXGRID).
2. Setting the width of various columns using grid units. On these options, a lightbulb link will appear nearby. Click it for a handy reference chart that shows the pixel equivalent of each grid unit value.

Freeform Grid: Think the pre-built grid configurations are too constraining? Then turn on the freeform grid and build your own dynamic grid! The freeform grid gives you the freedom to:

1. Specify your own site width, in pixels. This value will become MAXGRID throughout the theme.
2. Specify a gutter width, in pixels. This is the amount of space between each column.
3. Specify a width for each column of the site, in pixels.

With the freeform grid, you won't be working with grid units. It's up to you to make sure everything fits in each row! The sky is the limit, so have fun with it.

Please note: I am not the creator or maintainer of the 1KB CSS grid, nor am I affiliated with its creator. As of the time of this writing, I do not provide

technical support for modifying the 1KB CSS grid beyond the way it is implemented in the Citizen Press theme. If you'd like to learn more about the 1KB CSS grid, you can read the documentation at: http://www.usabilitypost.com/2009/05/29/the-1kb-css-grid-part-1/

3.3 Installation

Now that you have an understanding of templatebits and the grid system, let's move on with the quick-start:

1. Unpack the Citizen Press ZIP archive.
2. Connect to your site using your FTP client.
3. Copy the *citizenpress* folder to *wp-content/themes*.
4. Citizen Press MUST be used with a child theme. If you don't already have one installed, go back to your FTP client. Open the folder you just copied in Step 3. Find "*citizenpress-blankchild*". This is a blank starter child theme. Move "*citizenpress-blankchild*" into the root themes directory ("*wp-content/themes*"), the same directory where the "*citizenpress*" folder resides (Alternatively, you could use any child theme from http://www.pixelplanethemes.com/themes).
5. Now log on to your WordPress Dashboard. Go to Appearance > Themes. You should see both "Citizen Press" and "Citizen Press Blank Child" in the theme gallery. Activate "Citizen Press Blank Child".
6. You may now proceed to the Citizen Press Options panel to begin configuring your site.

That's all there is to it for installation! See the next section for configuration tips.

3.4 Typical Configuration

After you've activated a Child Theme as the default, you should see a new link called *Citizen Press Options* in the Appearance menu in the Dashboard (you

might see another options link depending on the Child Theme you activated). Go ahead and click on the *Citizen Press Options* link.

The options page is fairly straightforward and inline help is provided to help you work through the tabs. There are a lot of options, but don't feel overwhelmed – you don't have to set them ALL at once. As I said earlier, the options you don't use will be there should you need them later. Below, I've listed some of the tabs and other steps you might want to work through in a typical installation (step-by-step):

1. *Fill out or print a copy of the cheat sheets (Optional)* - This is optional but it may save you time and make it easier to visualize your setup. The cheat sheets (interactive PDFs) were provided in the *docs* folder in the Citizen Press ZIP archive. On these sheets you can jot down important categories, category IDs and templatebit locations. Either type your info directly on the interactive PDFs in Acrobat Reader, or print out the sheets and fill them out by hand. If you don't *have* the cheat sheets, you can download a copy from the online documentation at pixelplanethemes.com.
2. *Auxiliary Bar Tab* – If you decide to display the auxiliary bar at the top of the page, you can configure the settings in this tab.
3. *Masthead Tab* – This is where your site logo and Masthead widget area will appear if you decide to use them.
4. *Setup your Menus* – Step away from the options panel and go to the WordPress Menu Manager to set up your Main Menu and Footer Menu. You can also configure the Auxiliary Bar Menu and the Section Index Menu(s). Note: the Auxiliary Bar Menu and Section Index Menu(s) won't appear unless you've enabled them in their respective tabs in the Citizen Press Options (*Auxiliary Bar* and *Section Index*).
5. *Front Page Tab* – Return to the options panel, and click on the *Front Page* tab. Decide which tiers, news boxes, sidebars and other

templatebits should appear on your front page. Also, decide if you want to show the news ticker, the breaking news banner and the horizontal sidebar. **Tip:** you should make notes on the front page cheat sheet to help you remember where you placed things!

6. *Top Story & News Boxes Tab* - After you've determined where news boxes should be placed on your front page (Step 5), configure those boxes in this tab. If you wrote your category IDs on the newsbox cheat sheet, refer to it as you work through this step.
7. *Backup Your Settings So Far* – Don't lose what you've done so far! Go to the *Import/Export Options* tab and click "Export Settings" to download a text file of your settings so far. Save this file in a safe place incase you ever need to restore your settings later.
8. *Setup Sidebars in the Widgets Panel* – Step away from the options panel and add widgets to the sidebars you placed in Step 5. Refer to the front page cheat sheet to help you remember where you placed each sidebar.
9. *Feature Widget Tab (Optional)* – Return to the Citizen Press Options panel. If you included the Feature Widget in Step 5, configure it now in the Feature Widget Tab.
10. *Archive Sections Tab* – Set up the layout for your archive sections. Make notes on your archive page cheat sheet accordingly.
11. *Single Posts Tab* – Set up the layout and display options for single posts. Make notes on your single posts cheat sheet accordingly.
12. *Pages Tab* – Set up the layout and display options for Pages. Make notes on your page cheat sheet accordingly.
13. *Set up Archive, Posts and Pages Sidebar in the Widgets Panel (Optional)* – Step away from the options panel and add widgets to the Archives, Single Posts and Pages sidebars if you enabled them in Steps

10-12, respectively. Refer to your cheat sheets to help you remember where you placed each sidebar.

14. *Footer* – Return to the Citizen Press options page and customize your copyright message for the bottom of the page.
15. *SEO* – If you're NOT using a SEO plugin, you can set some basic SEO options here.
16. *Stats Codes* – If you have any tracking or statistics codes to add to your header or footer, add them here.
17. *Back-up your final settings* – You don't want to lose your hard work now, do you? Go to the *Import/Export Options* tab and click "Export Settings" to download a text file of your final settings. Save this file in a safe place incase you ever need to restore your settings later.

Other Tabs To Set At your Leisure:

1. *Grid* – If you'd like to change the grid settings from the default, you can do so here. You'll then need to go back to the *Auxiliary Bar, Masthead, Front Page*, *Archive Sections*, *Single Posts*, and *Pages* tabs to adjust the grid units for all of their respective columns so that they work with your new grid settings.
2. *Global* – Lets you configure some various site-wide settings, such as whether or not to show post excerpts or full content, post date format, excerpt length and more. Plus, instructions are given here for setting up a Favicon for your site.

Then just continue to play around with the settings until you get them the way you want. Adjust things to match the amount content you have on your site. For example, if you don't have many posts, you may not need to enable all three tiers on the front page.

After you've configured the basic layout of your site, you can either stick to the default design or create your own design. If you want to create your own design, read on!

3.5 Customizing the Design

First, an important word: All design customizations should be made using a Child Theme. **Never, ever, *ever* edit any file in the Citizen Press theme directory (wp-content/themes/citizenpress). Never add or remove any files to/from this directory.** If you edit any of the Citizen Press files directly, you'll lose your changes with future upgrades to the theme. Keeping the Citizen Press files untouched and doing all of your customizations by way of a Child Theme protects you from this! If you do decide to edit your Citizen Press Parent Theme files directly in spite of this recommendation, understand that you are doing so at your own risk. No support will be given for damage caused by editing the parent files.

Sorry to go off on that high horse, but it's very important! Now, let's discuss the possible design customization routes! There are several ways to customize the Citizen Press theme's design:

Citizen Press Colorific Child Theme – This child theme provides color and font pickers and it's available from pixelplanethemes.com If you set this as the default theme, you'll see a "Citizen Press Design Options" link in the Appearance Menu. Work through the tabs on the Design Options page to add your own colors/font preferences to the mix using the various color pickers. After you've settled on colors/fonts, be sure to back them up using the *Import/Export* tab in the Citizen Press Design Options Panel.

While you can achieve awesome, endless color and font combinations in Colorific, it doesn't let you change *every* aspect of design in the Color/Font editor. You won't be able to add fancy graphical backgrounds, padding, margins,

borders, etc, without going into the CSS file. The Colorific *style.css* overrides everything in the Design Options page.

Citizen Press Blank Child Theme – The Blank Child Theme comes with Citizen Press. It includes a stylesheet that imports styles from the Citizen Press stylesheet. Everything you place in this stylesheet will override the Citizen Press stylesheet. I suggest copying all the styles from the Citizen Press stylesheet and pasting them here if you want to change everything. (**Note:** Citizen Press uses a compressed stylesheet on the live site to improve loading time. A development stylesheet is provided with comments; the development stylesheet is what you should copy and paste your styles from since it's more legible!) The Citizen Press development stylesheet can be found at *citizenpress/style-dev.css*. Be sure not to edit the Citizen Press stylesheet directly. Instead, just copy everything and then paste it into the Blank Child Theme stylesheet. The development stylesheet is heavily commented to help you know what you're editing!

You can also start completely from scratch by removing the @import from the Blank Child Theme stylesheet.

Download a Child Theme – Over time, there will be other pre-designed Child Themes available at Pixel Plane Themes. These themes can serve as a starting point for custom designs as well.

Build your own Child Theme – See *Section 5* for instructions on building your own Child Theme!

Hire a web designer – Hiring a web designer to build a Child Theme or to customize an existing Child Theme could be another option.

3.6 Adding Custom / Plugin Content

You'll likely want to add plugins or your own custom content at some point. Many plugins will instruct you to add a snippet of code to some file in your theme. But since you're not supposed to edit the Citizen Press theme files, you'll add these code snippets by way of action hooks instead. If you've worked with a

WordPress theme framework before, this process will be familiar to you. (Citizen Press is packed with action and filter hooks. They are explained in more detail in Section 6).

This involves opening up the *functions.php* file of your Child Theme and writing a simple function to add the plugin code. Typically, the procedure is this:

1. You find and install the plugin you want. Let's pretend it's a featured content gallery. The plugin tells you to add code to your theme where you want the gallery to appear.
2. You want the gallery to appear on the front page, just below the main menu. So you look through the Citizen Press Function Directory (which you can find in the online documentation) and find an action hook that runs in the location you want the gallery to display. For example, *cpress_before_tier_1()* runs just before Tier 1 on the front page and just below the main menu.
3. Open up your Child Theme's *functions.php* and write a function like this (placing it AFTER the first *<?php* and BEFORE the last *?>* in that file):

```
function childtheme_featured_content_gallery() { ?>
....GALLERY PLUGIN CODE WOULD GO HERE...
<?php } add_action('cpress_before_tier_1',
'childtheme_featured_content_gallery');
```

4. Save the file and reload your site. If all went well, you should see your new content!
5. Repeat this process whenever you want to add plugin code or any type of custom content.

You don't need to fully understand the concept behind action hooks in order to use/write functions like the one above (if you want to understand the concept, see Section 6 of this guide). Here are just the essentials of what you need to know for every function you write like this:

1. The part in **blue** is the name of the custom function you're writing that adds the plugin code. You can use any name you want, just make sure it appears those two times and that the names match. The convention is to use the Child Theme's name in combination with a word or phrase that describes the type of content this function is adding.
2. The part in **black** is the name of the action hook. It must appear in that location in the code. Whatever hook you use here must come from the Citizen Press Function Directory.
3. The part in **green** is your custom code. It can be a mixture of HTML and PHP. It can contain as little as just the plugin code, or as much as your own HTML tags in addition to the plugin code. For example, you could add some 's or <p>'s, etc. *Just be mindful of the opening and closing PHP tags. You need a ?> wherever PHP ends and HTML begins, and you need a <?php wherever HTML ends and PHP begins.*
4. The part in **red** is just the required PHP code. Make sure it's all there or you'll get errors!

That's all there is to it. Editing hooks is where most of the power of Wordpress theme frameworks comes from. If you really learn your way around the hooks system, you'll find that you have plenty of flexibility beyond the options panel. See Section 6.

4 CITIZEN PRESS FILE STRUCTURE

This section quickly reviews the Citizen Press theme file structure. You'll never need to *modify* any of these files, but it's still good to have some idea of how the theme is structured. Everything described here is current as of Citizen Press Version 1.0. I'm only going to give an overview; if you want more details you can browse the files on your own when you get a chance – they're heavily commented! **Note:** you'll find a blank "*index.html*" file in every folder. This is for security purposes – it prevents the contents of a folder from being listed in the browser if that folder is accessed directly).

4.1 The "citizenpress" folder

This is the Citizen Press root folder. Typical WordPress theme files are here, such as *index.php, single.php* and *functions.php*. In Citizen Press, these files contain mostly function calls.

You'll also find the stylesheets in this folder. "style.css" is the main stylesheet in use on the Parent Theme (the "style.css" file in the Citizen Press root folder imports this stylesheet). "style.css" has been compressed to reduce file size and as a result it's not very readable. If you want to copy styles into a Child Theme, copy them from the "style-dev.css" file, which contains the exact same styles as "style.css", only it's uncompressed and heavily commented (thus much easier to read). The "style-dev.css" file is not loaded in the live site. Finally, this folder contains stylesheets for Internet Explorer 6 and 7.

The "README" folder contains documentation and references, and the "citizenpress-blankchild" folder is a blank child theme to help you start your child theme development.

4.2 The "images" folder

This is where all theme-related images go.

4.3 The "library" folder

The library is where everything happens! Here's an overview of the folders and sub-folders you'll find within:

- **js:** This folder contains all the JavaScript used for the site and for the options panel.
- **languages:** This folder contains language files. If more translations are added in the future they'll go here.
- **options_page:** This folder and the sub-folders within contain everything to do with the theme options panel.
- **templatebits:** This folder and its subfolders are all templatebit files. Templatebit files mostly consist of function calls.
- **theme_functions:** This is where most of the "meat" is. All of the functions called in the templatebit files may be found within the theme_functions files. The organization is pretty self-explanatory. Plus, there is a file called "*vars.php*" that contains global variables and constants used throughout the theme.
- **widgets:** This folder contains custom widgets that come with the Citizen Press theme. See *Section 10, Citizen Press Widgets, Issues & Templates* for more info.

5 WORKING WITH CHILD THEMES

5.1 The Parent-Child Relationship

There are many different ways to visualize the Parent/Child Theme relationship, but I like to think of the Parent Theme as a "face", and the Child Theme as a "mask" that is placed over the face. Masks can cover all or part of a face. Whatever part the mask *doesn't* cover, the original face shows through.

In the Parent/Child relationship, a Child Theme is always set as the default. WordPress then loads the Child Theme first, meaning the Child Theme's stylesheet and functions files override that of the Parent Theme. The Child Theme can use any of the Parent Theme's CSS, hooks (action and filter) and other functions as designated by the creator of the Parent Theme.

Any theme can be a Parent or Child Theme. Theme frameworks make the best "parents", however, because they contain elements such as hooks and pluggable functions that Child Themes can use for customization.

5.2 Creating a Citizen Press Child Theme

To build a Child Theme from scratch, here are the steps you'd take! If you already know how to do this, you can skip this section. ;)

1. Somewhere on your local computer, create a new folder. This will be your Child Theme's folder. Give it a name.
2. Next, open your favorite source code editor. Create a blank *functions.php* file and save it in the folder you created in Step 1.
3. Now, create a *style.css* file and save that in the folder from Step 1 as well.
4. Keeping the *style.css* file open, paste this code at the top, filling in your info accordingly (*code continues to next page*):

```
/*
Theme Name: Your Theme Name
Theme URI: http://yourthemeurl.com
Template: citizenpress
Description: Description of your theme
Author: Your Name
Author URI: http://yourwebsite.com
License: GNU General Public License v2.0
License URI: http://www.gnu.org/licenses/gpl-2.0.html
Tags: your, tags, here
*/
```

```
/* Import styles from the parent theme */
@import url("../citizenpress/style.css");
/* Add custom styles below this line */
```

The most crucial thing to note here is **Template: citizenpress**. This line tells WordPress that this theme will be a Child Theme of Citizen Press. The second crucial thing to note is the **@import** statement. This tells WordPress to import the default styles from Citizen Press. It's recommended that you leave this line in place unless you really want to start with no styles! *(Tip: you can also copy all of the styles from the Citizen Press development stylesheet and paste them into this stylesheet. If you do that, you can either remove the @import or keep it, it's up to you. The Citizen Press development stylesheet can be found at citizenpress/style-dev.css).*

5. Next, open your image editor and create a 300px by 225px screenshot graphic of your Child Theme. If you don't have an actual screenshot, a simple graphic with the name of your Child Theme will do. Save this graphic in the folder from Step 1 as *screenshot.png*.
6. Now you have the minimum files you need for a Child Theme. If your Child Theme will have shiny graphics, you should make an "images" folder. Go ahead and upload your new Child Theme folder to *wp-content/themes*. Then set this new theme as the default.
7. Now you can create your own design by replacing the default CSS with your own in *style.css* (below the @import statement if you left that line in there), and by utilizing hooks and pluggable functions in *functions.php* (See Section 6 for a rundown on hooks and pluggable functions). You can create your own template files if needed, such as *single.php*, *category.php*, etc. These files can use any of the Citizen Press hooks/functions.
8. *Optional:* in your *functions.php* file, you can also redefine the default variables and constants (for a list, see the online documentation). Finally,

it's possible to copy templatebit files from the Parent Theme to the Child Theme and/or to create your own templatebits (See Section 7).

6 HOOKS AND PLUGGABLE FUNCTIONS

This section focuses on advanced customization for Citizen Press using hooks and pluggable functions. You can perform most advanced customizations thanks to the WordPress Action and Filter hooks system. Citizen Press contains many action and filter hooks, and you can find a list of them in the Function Directory in the online documentation. Pluggable functions were introduced in WordPress 3.0, and they involve writing your own versions of existing functions to override default features. A list of Citizen Press pluggable functions can also be found in the Function Directory. I assume that all readers of this part of the guide are comfortable with WordPress, Child Themes, XHTML/CSS, FTP and at least basic PHP. If this doesn't describe you, you may wish to spend time learning those before diving into this section.

If you've worked with WordPress theme frameworks before, you may find that you already know the material described in the upcoming sections. In that case, you can skip this section.

6.1 Action Hooks

Action hooks are empty functions that serve as placeholders that you can use to insert custom code into various parts of the theme. WordPress itself contains many action hooks. Two that you might be familiar with are wp_head() and wp_footer(). Plugin authors often use them to run their plugin code (this is why many plugins break in themes that don't include wp_head() or wp_footer()). There are also action hooks that run on the edit post screen, in admin screens and in other locations in the WordPress interface. If you know what these hooks are and where they run, you can add your own code to them to make WordPress do some remarkable things.

Citizen Press contains plenty of action hooks that cover all parts of the theme, so you have many spots where you can drop custom code. You can find all of the action hooks in the Function Directory in the online documentation.

How do action hooks work? Think of action hooks as being little "key hooks" that have been placed all over the theme in various locations. wp_head() is one such key hook. wp_footer() is another. Whenever the web server executes a WordPress theme file, line by line, it will eventually run into one of these key hooks. If the key hook is empty, the web server will still acknowledge its existence but nothing will show up on the website for it because an empty key hook is…well…empty.

If the web server encounters a key hook with a *key* hanging from it, however, it will acknowledge both the key hook *and* the key before continuing on with the rest of the script. If the key has something cool to add to the website, such as a featured content gallery, this is where the web server will display the gallery and continue on with the script. If there are more than one keys dangling from a single hook, each key will be acknowledged in the order that they appear. If keys are to be acknowledged in a specific order, they'll have numbers on them…

Now, let's apply this little analogy to the code. The "key hooks" are the empty functions, as I stated before (wp_head(), wp_footer()). The "keys" are also functions, but they are functions that are "*hooked"* to the empty "key hooks". When you work with action hooks, you will be writing the custom functions (keys) that contain custom code (such as "add-this-line-to-your-theme" code from a featured content gallery plugin), and, using a WordPress function called add_action(), you'll hook those functions to an action hook (key hook). Your custom functions will then run whenever that action hook runs. It's as simple as that!

As I've said before, Citizen Press, like most theme frameworks, contains many of these action hooks. All of the built-in templatebits, in fact, are added to

the theme via action hooks. The Citizen Press source code has been heavily commented so that every single action hook is labeled, and you know which *keys*, if any, have been attached to each hook by default. If you're interested, I recommend you take some time to look over the theme files to get a feel for how the hooks have been setup. These hooks were added to allow people to add their own code, be it plugins or something of their own creation.

Writing a "key" function and "hooking" it to an action hook. To write a function that adds custom code to your theme, follow these steps:

1. What part of the theme are you adding the code to? Once you've determined that, search the Citizen Press Function Directory for a hook that runs in the location where you want to add the custom code.
2. Next, open the *functions.php* file of your Child Theme. Let's create the function that adds the custom code. In this example, I'm going to add a full-width extra row above the Auxiliary Bar at the top of the page.

```
function insert_row_above_aux_bar() { ?>
	<div class="row">
		<div class="column grid_<?php echo MAXGRID; ?> ">
			<p>I'M A NEW ROW ABOVE THE AUXILIARY BAR!</p>
		</div>
	</div> <?php }
```

   Now the "key" function is written. I wanted the new row to fit in with the grid, so I used the grid markup. Note: MAXGRID is a constant; it stands for the highest possible number of grid units, depending on what grid configuration has been chosen in the Citizen Press options.
3. Now all we have to do is "hook" this function to an action hook using add_action(). The action hook cpress_auxbar_before() runs just before the opening div of the Auxiliary Bar. So, just below the function I wrote in Step 2, I'd put this code:

```
add_action('cpress_auxbar_before', 'insert_row_above_aux_bar');
```

Putting it all together, it looks like this:

```
function insert_row_above_aux_bar() { ?>
	<div class="row">
		<div class="column grid_<?php echo MAXGRID;?>">
			<p>I'M A NEW ROW ABOVE THE AUXILIARY BAR!</p>
		</div>
	</div>
<?php }add_action('cpress_auxbar_before', 'insert_row_above_aux_bar');
```

4. Save the file and reload. You should see the new div above the Auxiliary Bar! Of course, you'd need to spice it up with some CSS, but at least now you can see how this system works.

Whenever you need to add something to the Citizen Press theme code, these are the steps you would take. In some cases you will need to use a different type of hook -- the filter hook -- discussed later on.

Multiple functions "hooked" to one action hook. Sometimes, you may have more than one function hooked to a single action hook. In that case, the functions will be executed in the order that they appear in the *functions.php* file. If you want to impose a specific order on them, however, then you'd simply stick on a number at the end of the add_action() function that specifies where that function falls in the queue. Take a look at this example:

```
function insert_row_above_aux_bar() { ?>
	<div class="row">
		<div class="column grid_<?php echo MAXGRID;?>">
			<p>I'M A NEW ROW ABOVE THE AUXILIARY BAR!</p>
```

```
            </div>
      </div>
<?php }
add_action('cpress_auxbar_before', 'insert_row_above_aux_bar', 1);

function insert_second_row_above_aux_bar() { ?>
      <div class="row">
            <div class="column grid_<?php echo MAXGRID;?>">
                  <p>I'M A SECOND NEW ROW ABOVE THE AUXILIARY BAR!</p>
            </div>
      </div>
<?php }
add_action('cpress_auxbar_before', 'insert_second_row_above_aux_bar', 2);
```

Notice the "1" and "2" that have been added to the end of each add_action(). The lower the number, the earlier the function runs. If I wanted the second function to run first, I'd simply swap the numbers. If I wanted a function to run last no matter what, I might set it to an incredibly high number, like 99.

"Removing" a function that is hooked to an action hook. Some action hooks in Citizen Press already have functions hooked to them. You might want to change those functions or remove them altogether. In that case, you'd use the remove_action() function to first "unhook" whatever "keys" were dangling there by default, and then write your own function to replace what you removed. For example, let's say you want to replace the Citizen Press Main Menu with a menu of your own:

1. First, we have to "unhook" the function that displays the default Main Menu. Every built-in templatebit in Citizen Press is loaded through a hook. You can find the hooks and hooked functions by searching the Function Directory in the online documentation. The function that displays the default Main Menu is called *cpress_show_default_main_menu()*; and it's hooked to the *cpress_main_menu()* action hook. Now open your Child

Theme's *function.php*. Type this code:

```
function unhook_main_menu() {
	remove_action('cpress_main_menu','cpress_show_default_main_menu');
} add_action('init ' , 'unhook_main_menu ');
```

Important: If the function you're removing has a priority number assigned to it, you must also include that number when performing remove_action(). Here is the basic pattern to follow:

```
function my_function_name() {
	remove_action('action_hook_name','function_to_remove',
priority_number);
} add_action('init ' , 'my_function_name');
```

2. Save the file and reload your site. The Main Menu should be gone! Now, go back to *functions.php* and write a function to add your own custom menu. Hook this function to cpress_main_menu() :

```
function new_main_nav() { ?>
	<!—-PLACE YOUR NEW MAIN MENU CODE HERE --!>
<?php }
add_action('cpress_main_menu', 'new_main_nav');
```

3. Save the file and reload your site. Your new menu code should show up where the old menu once was!

Action hooks are future-proof. Since you're only editing the *functions.php* file of your Child Theme when you write action hooks, you're manipulating the Citizen Press Parent Theme without even touching its core code! This means that you can add code to Citizen Press without worrying about messing anything up.

More importantly, if you upgrade Citizen Press in the future, you won't lose your customizations!

6.2 Filter Hooks

Filter hooks are a second type of hook in WordPress, only these aren't exactly placeholders in the same sense that action hooks are. Filter hooks allow you to write functions that *change* some aspect of an existing block of content. In a sense, you are *filtering* out what you don't want and replacing it with what you do want. Analogy time! Imagine walking into your living room and seeing a huge "EDIT" button hovering over each item of furniture. You press the button that's hovering over the sofa. You proceed to change the pattern, size and material; plus you add a cat. After "saving" your changes, you still have a sofa, but you've given it a new look (and a new cat). Next, you move on to the table in front of the sofa. It's got plates on it. You like everything about the table except for the plates. So you press the EDIT button above the table and remove the plates. Finally, there is a large window on the wall opposite the sofa. You decide you don't want a window there. So you press the window's EDIT button and remove the window altogether. Voila, window's gone!

This is the basic concept behind filter hooks and how they differ from action hooks. With action hooks, think *key hooks* (a.k.a. empty functions), outputting the code of whatever functions happen to be attached to them. With filter hooks, think of a block of code (a function) that already exists, but imagine that this block of code is "editable", meaning you can attach a filter to that block that modifies the contents within.

How do filter hooks work? WordPress itself contains plenty of filter hook functions. The function that outputs a "[…]" at the end of post excerpts is a filter function. If you've ever tried to remove the "[…]", you had to write a function that

filtered that particular part out, just like you removed the plates from the table in the living room example earlier.

Filter hook functions store all of the output in a single variable, which is then returned at the end of the function. Going back to the living room then, all of the couch's output may be stored in a variable called $couch. This variable would then be returned by a function called show_couch().

Many elements within the Citizen Press theme are "filterable." Elements like post and page titles, post metadata and the page headers are all loaded using filter hook functions, and you can change them by writing your own filter functions. With action hooks, we used the WordPress add_action() function to hook a custom function onto an action hook. With filter hooks, we use the WordPress add_filter() function to hook a custom filtering function onto a filter hook.

Example of a filter hook function. To get an idea of a how a filter hook function looks, take a look at the function below. This function, cpress_show_post_author_byline(), outputs the post author bylines in Citizen Press:

```
function cpress_show_post_author_byline() {
global $authordata;
	$byline .= '<span class="author">';
	$byline .=  __("By" , "citizen_press") . ' <span class="author_name">';
	$byline .= '<a href="';
	$byline .= get_author_posts_url($authordata->ID, $authordata-
>user_nicename);
	$byline .= '" title="' . __("View all posts by" , "citizen_press") .
get_the_author() . '">';
	$byline .= get_the_author();
	$byline .= '</a></span></span>';
echo apply_filters('cpress_show_post_author_byline', $byline);
}
```

Notice how all of the output is stored in a single variable called $byline. The "apply_filters()" function near the end of the hook echoes this variable, which causes the title to display. You won't be working directly with the filter function itself, however.

Modifying the output of a filter hook function. Whenever you want to modify the output of a filter hook function like the one shown above, you'd follow these steps:

1. Decide what part of the theme you want to change. Then you need to find out if that part of the theme can be changed using filters. The online Citizen Press Function Directory lists all of the filterable functions and their associated variables. For the sake of this example, let's say you want to wrap a red container around the author bylines. So you'll be modifying the cpress_show_post_author_byline() filter hook function above. It stores all of its output in the variable $byline.
2. Open the *functions.php* file of your Child Theme. First, you need to write the custom function that adds the extra :

```
function my_custom_byline_function($byline) { ?>
       return '<div style="background: #ff0000;">' . $byline . '</div>';}
```

   By passing the $byline variable as a parameter of your custom function, you can modify everything contained in that variable as a whole. Everything contained in the $byline variable is the entire post byline. So this function adds the red container div around it. You're not done yet, however…
3. We have this function written but it doesn't have any effect on your site yet. You still need to turn it into a filter and attach it to the filter hook function using add_filter(), like so (the bold indicates what was added):

```
function my_custom_byline_function($byline) { ?>
	return '<div style="background: #ff0000;">' . $byline . '</div>';
}
add_filter('cpress_show_post_author_byline','my_custom_byline_function');
```

Go ahead and save your file and refresh your site. You should see a red container div around all of your bylines! Of course you'll want to make your actual modifications look prettier, but at least you can see the basic idea behind filter hooks!

Replacing the output of a filter hook function. You can also replace everything inside a filter hook function with entirely different content of your own. Remove the function we wrote in the previous example, and replace it with the function below. This one replaces the post author bylines with a smiley face:

```
function my_smiley_byline_function() {
	echo ':)';
}
add_filter('cpress_show_post_author_byline','my_smiley_byline_function');
```

Now all you'll see instead of bylines is a smiley face! Notice that we don't need to pass the $byline variable this time because we're replacing everything in the variable. Of course, you'd probably want to replace the default bylines with something more practical than a smiley face!

Remember how we removed the windows in the living room example? We can remove the bylines altogether as well. Replace the code above with this:

```
function my_no_byline_function() {
	echo '';
}
add_filter('cpress_show_post_author_byline','my_no_byline_function');
```

Now reload your site. You should see no bylines at all!

Filter hooks are future-proof. It's same deal as with action hooks – when you work with filters, you are only touching the Child Theme's *functions.php* file, and therefore protecting your customizations from future updates.

6.3 Pluggable Functions

Some parts of Citizen Press cannot be customized directly using action and filter hooks. They are instead controlled by functions that are *pluggable*. When WordPress encounters a pluggable function, it first checks to see if that function exists in the Child Theme. If it does, WordPress will use the Child Theme version. If the function doesn't exist in the Child Theme, WordPress will fallback on the version of that function defined in the Parent Theme. Pluggable functions may exist inside of action hook or filter hook functions, but to edit them directly, you must rewrite the pluggable function in your Child Theme. This is best illustrated with an example:

```
if ( ! function_exists( 'cpress_author_info_box' ) ) :
function cpress_author_info_box() { ?>
	<!--Post Author Info -->
	<div class="post_author_info_container">
		<span class="alignright"><?php echo
get_avatar(get_the_author_meta('user_email'), 55, none, 'User Avatar');
?></span>
	<h4 class="post_author_info_header"><?php _e('About the Author | ',
'citizen_press');?> <?php the_author_posts_link(); ?></h4>
	<p><?php the_author_meta('description'); ?></p>
	<br class="clearFloat" />
	</div><!--Post Author Info-->
<?php } endif;
```

The above function, cpress_author_info_box(), creates the "About the Author" box that may appear after posts and/or pages. Notice that the function is wrapped inside a conditional statement. The first line says, in plain English: *if the*

function called cpress_author_info_box() does not exist in the Child Theme, do the following: [default function here].

After the end of the function, don't forget to include "endif;" to close the conditional.

Creating your own version of a pluggable function: It's really easy to work with pluggable functions:

1. Search the Citizen Press Function Directory for a pluggable function to modify. Make note of the function name.
2. Open your Child Theme's *functions.php* and write a new function with the same name as the function you identified in Step 1. You can put anything you want in the body of the new function. **DO NOT include the "function exists" conditional. Just write it out as a normal function.**
3. Save the file and visit the appropriate page of your site. You should see the changes!

7 MODIFYING/CREATING TEMPLATEBITS

We've discussed Child Themes, hooks and pluggable functions. Now it's time to talk more about templatebits. First off, if you haven't read the description of templatebits earlier in this guide (Section 3.1), you should do so before reading this section.

7.1 Modifying a Default Templatebit File

Most of the time you can modify templatebits using hooks and pluggable functions. However, in some cases you might want to modify an entire templatebit file.

When loading templatebit files, Citizen Press first checks to see if they exist in the Child Theme. If not, it'll load the version from Citizen Press. This means that if you wanted to do your own thing with a default templatebit file (beyond modifying hooks and pluggable functions), you can simply create a custom version of that file in your Child Theme directory and the custom version will be used instead. Here are the steps:

1. The directory structure of the Child Theme needs to match that of the Citizen Press theme. Logon to your site via FTP. Navigate to your Child Theme directory. Make a *library* folder in your Child Theme's root directory. Then, inside that *library* folder, make a *templatebits* folder. Inside the *templatebits* folder, make a *built-in* folder. Double-check these folders and sub-folders, they must be named and nested exactly as described or it won't work.
2. Now, navigate to *wp-content/themes/citizenpress/library/templatebits*. Download the templatebit file(s) you want to modify. If you download any files from the *built-in* folder, separate them somehow from the other files.
3. Return to your Child Theme's *templatebits* folder. Upload the files you just downloaded. **Be careful:** if any of the files you downloaded were from the *built-in* folder, be sure to upload them into the Child Theme's *built-in* folder. Do **NOT** change the file names!
4. Now open the newly uploaded files and make your desired edits. You can still use any of the Citizen Press hooks/functions. Reload your site and view your changes. If you get file not found errors, check to see that *1) you created the folders correctly in Step 1; 2) you uploaded the files correctly in Step 3*.

7.2 Creating Custom Templatebits

It's also possible to create custom templatebits. Procedures are listed below. Remember, *modular templatebits* are those that users can place using the Citizen Press options panel (things like newsboxes and generic sidebars). *Built-in templatebits* (things like menus, Auxiliary Bars, and page-specific sidebars) already have pre-defined locations in the theme (but can usually be moved around using action hooks).

Modular Templatebits:

1. The directory structure of the Child Theme needs to match that of the Citizen Press theme. Logon to your site via FTP. Navigate to your Child Theme directory. Make a *library* folder in your Child Theme's root directory if one doesn't already exist. Then, inside that *library* folder, make a *templatebits* folder if one doesn't exist. Double-check these folders; they must be named and nested exactly as described or it won't work.
2. Create your templatebit file(s). They must be PHP files. Upload them to the *templatebits* folder you created in Step 1.
3. Now you need to tell Citizen Press that the new templatebits exist. Child Theme templatebit names are stored in an array called $childtbits. Open your Child Theme's *functions.php* file. Include this line:

```
$childtbits = array("filename1","filename2");
```

Replace the words "filename1" and "filename2" with the actual filenames of your new templatebits. **Do NOT include the PHP extension** – the script will add it later on. If you have more

than two templatebits to add, just add them on to the end of the list after "filename2".

4. After saving *functions.php*, visit the options panel. Add one of the new templatebits in a location that accepts templatebits. (Also, click on the "List of Filenames" lightbulb link. Verify that the new templatebits show up in the list).
5. Reload your site and see the new templatebit in action. If it doesn't work, check to see that *1) you created the folders correctly in Step 1; 2) you uploaded the files correctly in Step 2*; *3) you typed the $childtbits array name and new templatebit filenames correctly in Steps 3-4.*
6. If you add more modular templatebits in the future, add their filenames to the $childtbits array in *functions.php* after uploading them to the Child Theme *templatebits* folder.

Built-in Templatebits:

1. The directory structure of the Child Theme needs to match that of the Citizen Press theme. Logon to your site via FTP. Navigate to your Child Theme directory. Make a *library* folder in your Child Theme's root directory if one doesn't already exist. Then, inside that *library* folder, make a *templatebits* folder if one doesn't exist. Inside the *templatebits* folder, make a *built-in* folder if there isn't one. Double-check these folders and sub-folders, they must be named and nested exactly as described or it won't work.
2. Create your templatebit file. It must be a PHP file. Upload it to the *templatebits/built-in* folder you created in Step 1.
3. Now you need to include the new templatebit file in the theme using the cpress_include() function and action hooks:
    i. First decide where this new templatebit should go.

ii. Then search the Citizen Press Function Directory for an appropriate action hook.

iii. In the Child Theme's *functions.php* file, write a function that includes the new templatebit file, and attach that function to the action hook. Here's an example:

```
function my_new_tbit() {
   cpress_include('filename', 'built-in');
   } add_action('action_hook_name', 'my_new_tbit');
```

Replace the function name ("my_new_tbit") with a more descriptive function name. Notice that the function name appears twice so change it twice. Next, replace "filename" (Line 2 in the code above) with the actual filename of your new templatebit. **Do NOT add the .php extension**. Finally, replace "action_hook_name" on line 3 above with the name of your chosen action hook.

4. Save and reload your site. Verify that the new templatebit shows up. If it doesn't work, check to see that *1) you created the folders correctly in Step 1; 2) you uploaded the file correctly in Step 2*; *3) you created the function correctly in Step 3 – make sure all of your file names and function names are correct*.

8 JAVASCRIPT AND CITIZEN PRESS

Moving right along, it's time to focus on JavaScript! Citizen Press provides the action hook *cpress_load_scripts()* for the purpose of loading custom JavaScript. (FYI: *cpress_load_scripts()* is hooked to *cpress_scripts()*, which is in turn hooked to *cpress_document_head()*, running before *wp_head()*).

8.1 Loading JavaScript Libraries with wp_enqueue_script()

Rather than load JavaScript libraries separately, Citizen Press makes use of the WordPress function *wp_enqueue_script()* to call in each library as it is needed by a particular script. This process is described in Section 8.2. WordPress comes packaged with a large number of script libraries, so it makes a lot of sense to tap into those. You can read more about *wp_enqueue_script()* and see a list of libraries that come with WordPress at http://codex.wordpress.org/Function_Reference/wp_enqueue_script.

8.2 Loading Custom JavaScripts – cpress_load_scripts()

You can use the *cpress_load_scripts()* action hook to load custom JavaScripts. By default, three functions are hooked to *cpress_load_scripts()*: *cpress_default_js()*, which loads the default JavaScript plugins used in the theme, like the tabbed sidebar code and the News Ticker; *cpress_default_ticker_config()*, which loads default JavaScript settings for the News Ticker and *cpress_comment_reply(),* which loads the script that moves the comment form below the comment you're applying to. You can remove either of these functions using remove_action() in a Child Theme. To load custom scripts using cpress_load_scripts():

1. Prepare the script file.
2. Upload it to your Child Theme's directory. It doesn't matter where, just as long as it's in the Child Theme directory.

In your Child Theme's *functions.php* file, write a function like the following that hooks to cpress_load_scripts():

```
function my_custom_scripts (){
      wp_enqueue_script('yourscript', '/wp-
content/themes/childthemefolder/path/to/yourscript.js', array('jquery'));
```

```
} add_action('cpress_load_scripts', 'my_custom_scripts');
```

Replace the bolded text with the appropriate information. Make sure the two red and two blue parts match, respectively. Note: it is within this function that you call in whatever library your script depends on. These are stored in an array. If you want to use a library other than jQuery, simply change "**jquery**" in the example above to the name of the other library, such as "prototype" or "scriptaculous". If your script relies on multiple libraries, for example, both jQuery and jQuery UI Tabs, your function might look like this:

```
function my_custom_scripts (){
   wp_enqueue_script('yourscript', '/wp-
content/themes/childthemefolder/path/to/yourscript.js', array('jquery',
'jquery-ui-tabs'));
} add_action('cpress_load_scripts', 'my_custom_scripts');
```

Notice how we added more script names in the array. For more detailed information on working with *wp_enqueue_script()*, see http://codex.wordpress.org/Function_Reference/wp_enqueue_script.

3. When you've got your function how you want it, save and view the source code of your site to verify that the script is loading.
4. If the script also involves HTML, you can use the other Citizen Press action hooks (See Section 7) to find a place to insert that HTML code. And edit your Child Theme's *style.css* to insert styles your script requires.

8.3 Changing the News Ticker JavaScript Settings

Citizen Press comes with a News Ticker (it's a built-in templatebit that appears on the front page by default if enabled) that cycles through post headlines from categories you set in the Citizen Press options panel. The ticker is powered by a jQuery plugin called "JCarousel Lite". JCarousel Lite comes with

quite a few settings and parameters that you can use to configure the functionality of the ticker, such as the speed, number of items, etc. You can't change these settings in the options panel, however, but you can do so by editing the JavaScript directly.

If you're not familiar with JCarousel Lite, I suggest visiting its website: http://www.gmarwaha.com/jquery/jcarousellite/. Instructions on modifying the settings can be found there. Without further ado, let's proceed.

1. First you need to prepare a JavaScript file on your computer containing your new news ticker settings. Either download a copy of the default Citizen Press ticker file (*wp-content/themes/citizenpress/library/js/tickersettings.js*) and change the settings within, or create a new file. Check the JCarousel Lite website for configuration options: http://www.gmarwaha.com/jquery/jcarousellite/.
2. Upload your new settings file to your Child Theme directory. Unlike custom templatebit files, you can put your settings file anywhere in this directory.
3. Open your Child Theme's *functions.php* file. Add this code

```
add_action('init', create_function( '$a',
"remove_action('cpress_load_scripts','cpress_default_ticker_config', 2);"
) );
function my_new_ticker_settings (){
	wp_enqueue_script('newtickersettings', '/wp-
content/themes/childthemefolder/path/to/newtickersettings.js',
array('jquery'));
} add_action('cpress_load_scripts', 'my_new_ticker_settings');
```

The first thing we do is remove the default ticker JavaScript settings. Then we write a function that loads your custom settings. This function is hooked to the *cpress_load_scripts()* function. Replace "path/to/newtickersettings.js" with the appropriate path and file name.

4. Save and view. The ticker should now have your new settings!

If you need to change the *appearance* of the News Ticker, you'll need to use CSS and also the associated News Ticker hooks. You can also modify the News Ticker templatebit file (instructions on using hooks are given in Section 6; and templatebit modification instructions can be found in Section 7).

Please note: I am not the author or maintainer of the jCarousel Lite plugin. As of the time of this writing, I do not provide technical support for jCarousel Lite beyond customizing its settings and incorporating it into the Citizen Press theme. Please visit the author's site at http://www.gmarwaha.com/jquery/jcarousellite/ if you have questions that fall out of this scope.

9 FORCING GRID SETTINGS

Next, we will talk about advanced customization of the CSS grid Citizen Press uses. The grid-related settings (grid configuration, overall site width, gutter width and column widths) in the Citizen Press theme are controlled by PHP constants that you can redefine in Child Themes. These elements are:

- Which grid configuration to use (if you want to override the configuration set in the options panel)
- Grid units of each column in the site (again, if you want to override the column widths set in the options panel)
- The overall site width (MAXGRID) if you're using the freeform grid configuration.

By default, these values are coming from whatever is set in the options panel. But if you want to force your own grid setting(s) and ignore what's in the options panel, you need to redefine the constants in your Child Theme. This is intended for advanced users who want more control over the widths of their site and

columns when building Child Themes but don't want to copy over a bunch of theme files.

You can find the column width constants on the "Column Width Constants" page in the Online Documentation. All you have to do is add some lines in your Child Theme's *functions.php* to redefine the one(s) you want to change. The procedures vary depending on whether or not you're forcing one of the pre-built configurations or the freeform configuration.

Forcing Grid Settings – Pre-built Grid Configuration. If you want to force the grid to use a specific pre-built configuration, you would first paste the following lines into your Child Theme's *functions.php* file:

```
define("FORCEGRID", true);
define("USEGRID", "9-80-20-900");
define("MASTHEAD_COL1", 5);
/*...more columns can go here ... */
define("FORCEGRID_WHAT", "Quick note listing the values you're overriding");
```

Now, change the bold items to match your preferences:

- FORCEGRID should be set to "true" (it's false by default). This causes Citizen Press to display a warning in the options panel, letting users know that you're overriding some (or all) grid settings.
- USEGRID should be set to whatever grid configuration you're planning to force. The four built-in values are: *9-80-20-900*, *14-60-10-980*, *10-80-20-1000* and *12-60-20-960*. (MAXGRID is the first number in each of those configurations). In the example, I've forced the grid choice to be the 9-80-20-900 configuration (a maximum 9 columns of 80 pixels wide, 20-pixel gutter and overall site width of 900 pixels; MAXGRID is 9 in this case), no matter what the user has set in the options panel.

- Now we'll start setting our own values for the columns. Go to the Online Documentation and visit the page called "Column Width Constants". Listed are the constants that control the column widths, one for each column of the site. Find the one you want to set the width for, and redefine it in your child theme's *functions* file. For example, in the code above, I had this:

```
define("MASTHEAD_COL1", 5);
```

  This forces Column 1 of the Masthead to be 5 grid units wide, no matter what the user has set in the options panel. Repeat this step for every column you want to change. Keep MAXGRID in mind when you're setting column widths!
- Finally, it's time to fill in FORCEGRID_WHAT. For this constant, type a list of everything you're forcing (you might also want to include the actual forced values. I did so in parentheses in the example below). If you're forcing ALL of the grid units, then just type "Everything". Whatever you type for FORCEGRID_WHAT will show up in the options panel in the warning message, so be as informative as possible:

```
define("FORCEGRID_WHAT", "Grid Choice (9-80-20-900), Masthead Column 1
(5)");
```

- You're done! Save your functions file and view your site to see the changes. Visit the options page to ensure that the warning message shows up.

Forcing Grid Settings – Freeform Grid Configuration. If you want to force the grid to use your own freeform grid settings, the process is similar to the one you'd follow for the pre-built configuration, with a few differences.

First, paste the following into your child theme's *functions.php* file:

```
define("FORCEGRID", true);
define("USEGRID", "freeform");
define("MAXGRID", "657");
define("FORCEGRID_WHAT", "Quick note telling what values you're overriding");
```

Obviously you're setting FORCEGRID to be true, to cause a warning to show up in the options panel. Then you'll set USEGRID to "freeform". Set MAXGRID to be the overall site width, in pixels. Only enter a number here, like I did in the example (I entered "657", meaning 657 pixels wide). Then, for FORCEGRID_WHAT, type out a message informing users what values you're forcing (put "Everything" if you're forcing everything).

Now you need to set the values! You'll do this using CSS. In the Online Documentation, search for the page called "Freeform Grid CSS Classes". On that page, you'll find a list of classes for you to copy over into your child theme's *style.css* file. Instructions are given to help you fill it out, but basically you will be setting column widths, the gutter width and the wrapper width.

Save your stylesheet and functions file, and then view your site to see the changes! Also visit the options panel to ensure that the warning message shows up.

10 CITIZEN PRESS EXTRA FEATURES

This last section touches briefly on a few "extras" that come with the Citizen Press theme. They are there to help spice up your site! More of these extras can be added in future versions depending on interest.

10.1 Issues Custom Taxonomy and Widget

In WordPress it's possible to create your own taxonomies to organize your content. Citizen Press includes a taxonomy called "Issues". You can group your

news stories into “Issues”. This is useful if you have a printed version of your paper and want all of the stories from a certain edition to appear together on the site, regardless of which category each story appears in. It’s also useful for “special issues” or series.

Issues are just like categories. You can create them either by clicking on the “Issues” link in the Posts menu in the Dashboard, or while typing a new post. You can have sub-issues as well. Assign a post to an issue just like you assign them to a category, on the post-editing page.

Issues show up in the Citizen Press archive page template (*archives.php*). There is also a widget called “Issues” which you can add to any sidebar. This widget will display your issues in a drop-down box. Future versions of the Citizen Press theme may expand more on the Issues concept.

10.2 Featured Posts, Featured Pages and Images Widgets

Citizen Press 1.0 comes with these three custom widgets. The **Featured Posts** widget lets you feature posts from a category. The widget lets you set a number of things, such as whether or not to show the post’s thumbnail (featured image), the size and alignment of the thumbnail, how many posts to show, how to order them and more. You can place this widget in any sidebar, and you can use as many of them as you like! Each individual instance of the widget has a unique CSS class, so you can give them unique styles. The **Featured Pages** widget is exactly the same as the Featured Posts widget, only it works with Pages instead of posts! These widgets are good for highlighting content that does not appear in the main news boxes.

The **Images** widget lets you display up to four images in two rows of two. This is useful for displaying ad blocks, social media icons, or anything else you can come up with. You can set the size of each individual image in the widget, plus optionally link them to an external (or internal) location. There are a lot of

settings in this widget, so spend some time playing around with it. You can have as many instances of the **Images** widget as you want.

10.3 Full-Width Page and Archives Page Templates

Citizen Press 1.0 includes two page templates. One is a full-width page (one column with no sidebars), and the other is a basic archives page. Use these templates if you want a Page without sidebars, or if you want to create an archives page. You'll find them in the "Templates" section when creating or editing a Page.

11 RESOURCES FOR FURTHER HELP

Well, that concludes this guide! For anything that was not covered here, search the online documentation at http://pixelplanethemes.com/. Because procedures and features are bound to change over time, the online documentation will always contain the latest instructions!