Hacking Wordpress

A Primer for PHP Programmers

Eli White

Vice President — One for All Events

@EliW

One for All Events

www.oneforall.events

Hacking Wordpress

A Primer for PHP Programmers

Eli White

Vice President — One for All Events

@EliW

One for All Events

www.oneforall.events

Why WordPress?

… and why should we listen to you?

In the beginning…

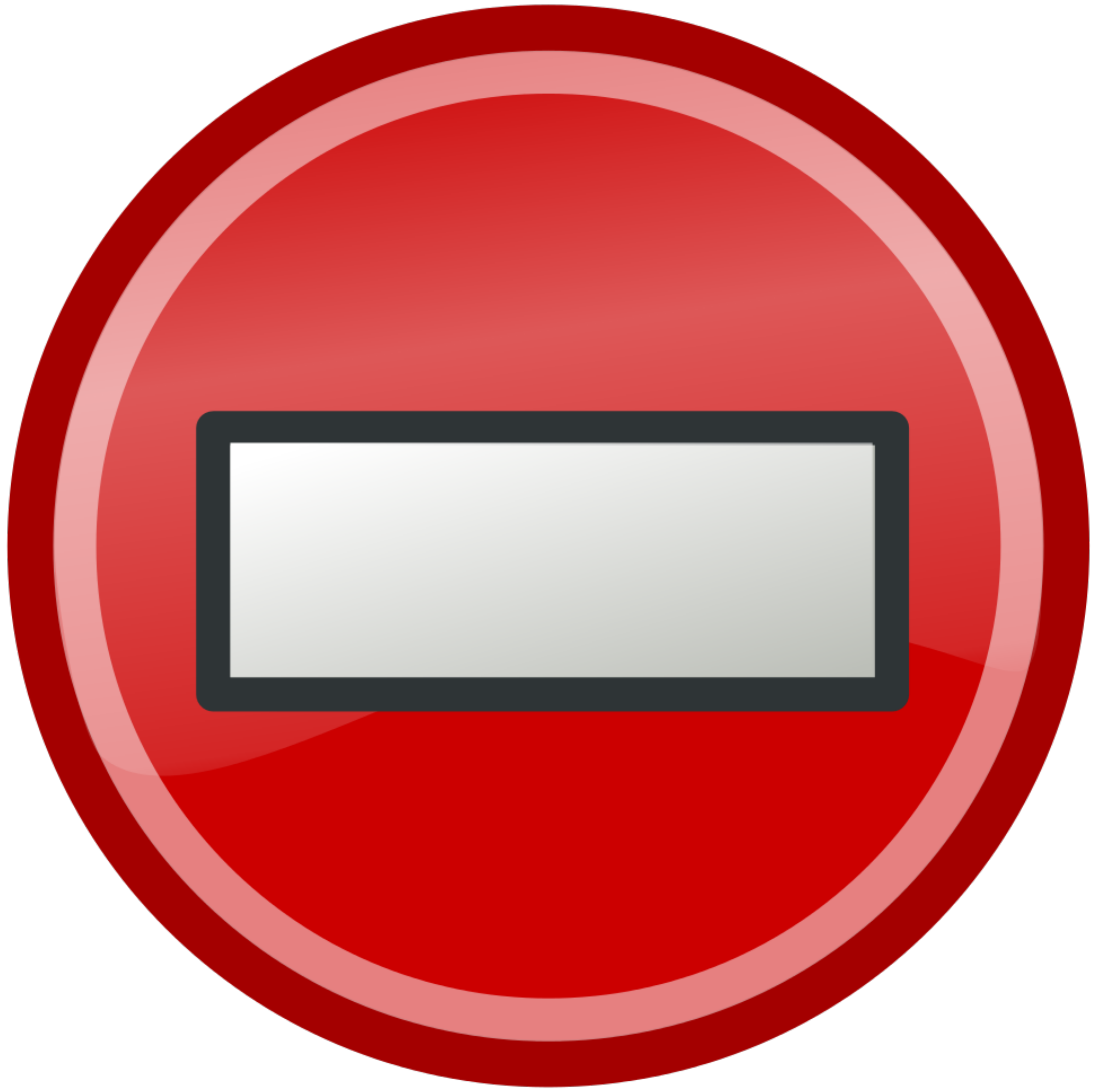
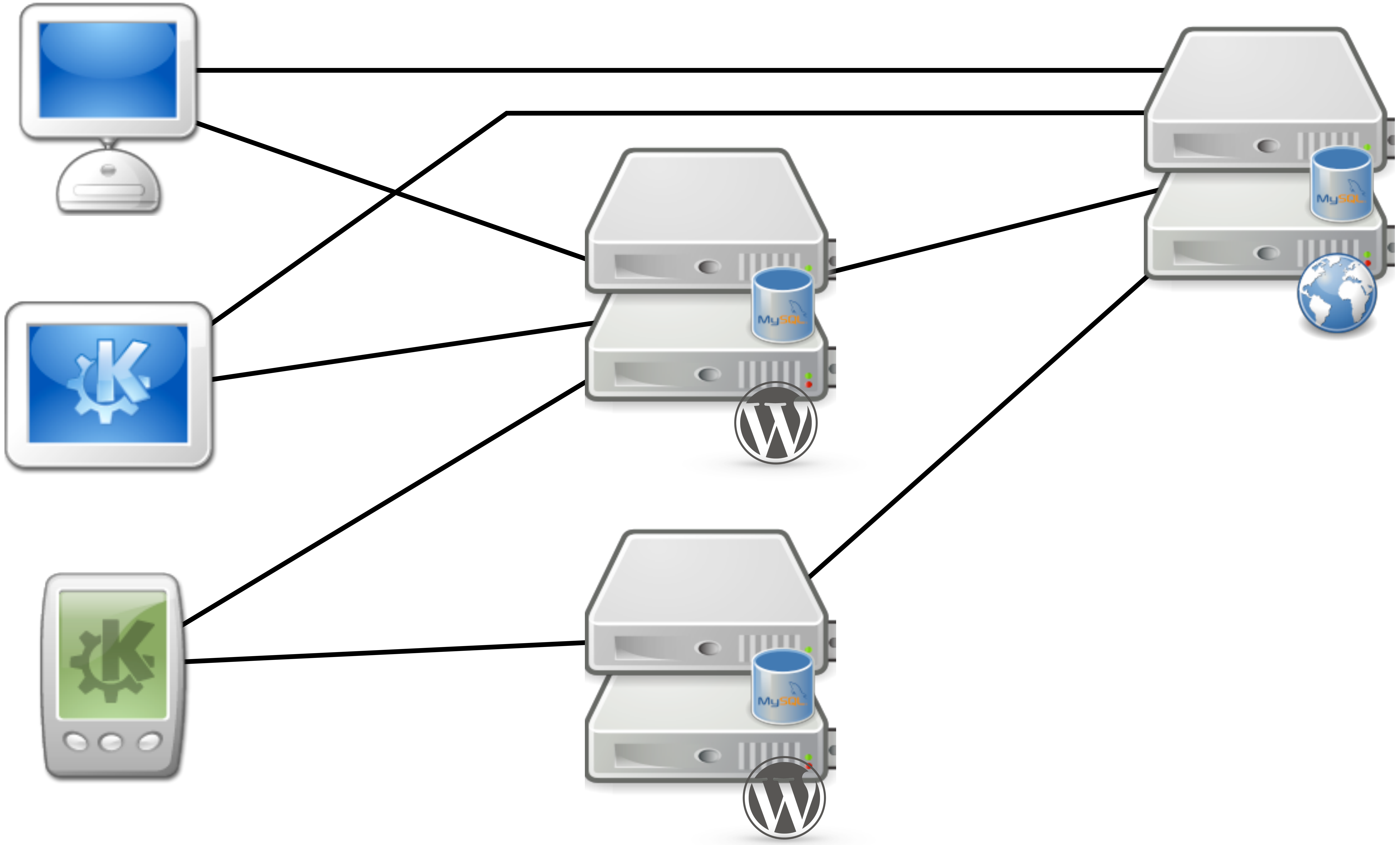


Image by RRZEicons [CC-BY-SA-3.0], via Wikimedia Commons

But then came…

php[architect] Infrastructure



Images by RRZEicons [CC-BY-SA-3.0], via Wikimedia Commons

Coding in WordPress

… as a Framework

How WordPress Does It

Really great documentation:

http://codex.wordpress.org/

Heaping help of functions / framework:

http://codex.wordpress.org/Function_Reference

https://developer.wordpress.org/reference/

Escaping Output

http://codex.wordpress.org/Data_Validation

`esc_html($text);`	Returns the text escaped for safe HTML output.
`esc_textarea($text);`	Encodes text to be safely used inside of a <textarea> element.
`esc_attr($text);`	Encodes text to be used safely inside of an HTML tag attribute.
`esc_js($text);`	Used to encode any inline JavaScript that you need to create & echo.
`esc_url($text);`	Sanitizes an URL to be output.
`esc_sql($text);`	Sanitizes user input being used in database queries, like PDO::quote()

Database Functions

http://codex.wordpress.org/Class_Reference/wpdb

Returns an array of objects

```
$rows = $wpdb->get_results("SELECT id, speaker, talk FROM event”);
```

Run arbitrary query

```
$wpdb->query('DELETE FROM event WHERE speaker = 42’);
```

Use prepared statements

```
$wpdb->query($wpdb->prepare('DELETE FROM event WHERE speaker = ?', $sid));
```

Helper methods to allow database updates without direct queries

```
$wpdb->insert($table, $data, $format);
$wpdb->replace($table, $data, $format);
$wpdb->update($table, $data, $where, $format = null, $where_format = null);
$wpdb->delete($table, $where, $where_format = null);
```

Return just a single row/column/data-point from a database query

```
$wpdb->get_row('query', output_type, row_offset);
$wpdb->get_col('query', column_offset);
$wpdb->get_var('query', column_offset, row_offset);
```

…and so much more

Plugins vs Themes

… two ways to add your own code

Themes

Your site's design

HTML & CSS

Plugins

Workhorse of WordPress

Real PHP code

Themes

~~Your site's design~~

~~HTML & CSS~~

Plugins

~~Workhorse of WordPress~~

~~Real PHP code~~

Same!

Themes

~~Your site's design~~

~~HTML & CSS~~

Meant to be 'specific' to your website

Plugins

~~Workhorse of WordPress~~

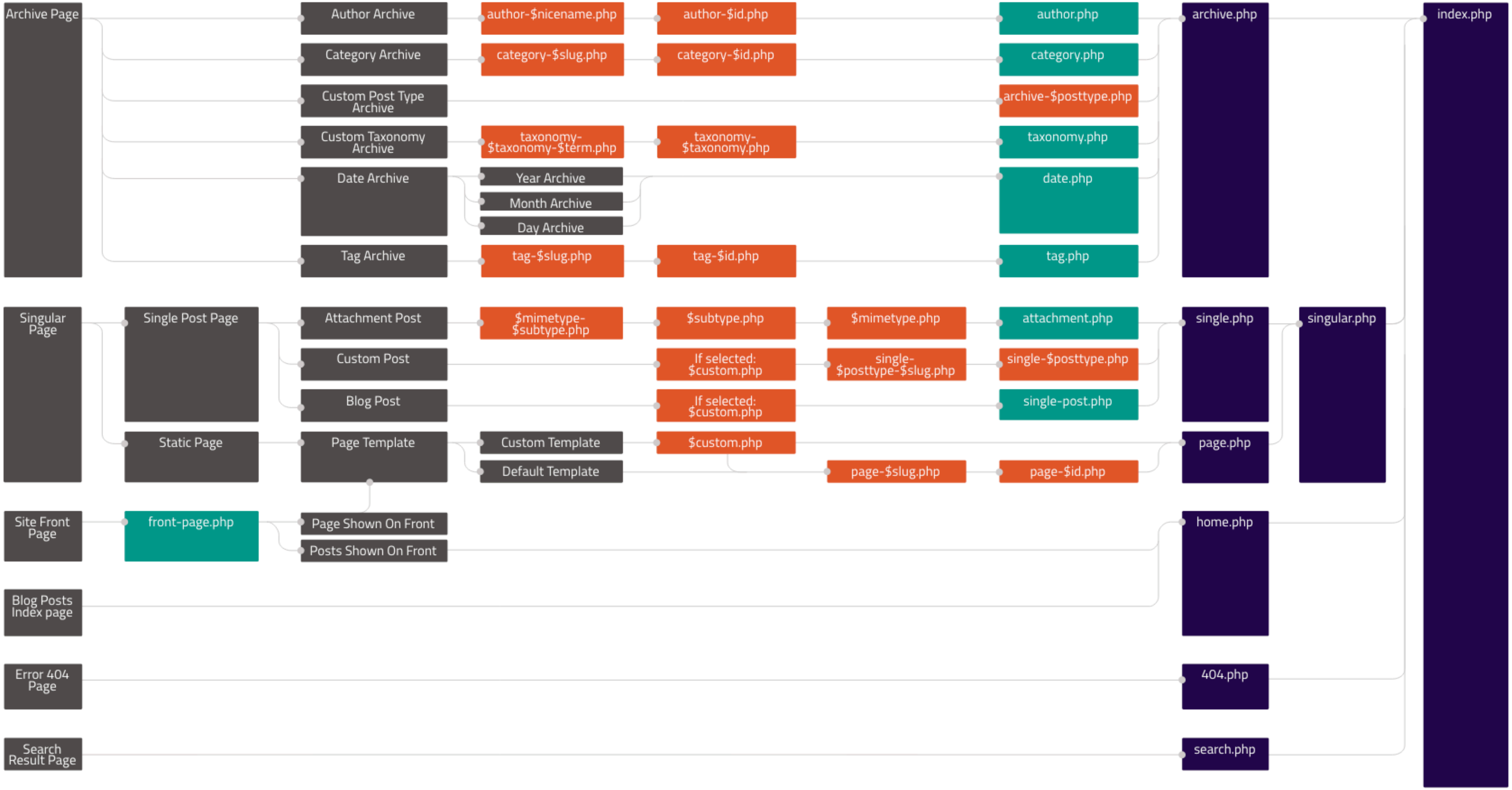
~~Real PHP code~~

Meant to be reusable between different sites.

Difference of Themes?

Code goes in file:

`functions.php`



Archive Page
Author Archive → author-$nicename.php → author-$id.php → author.php → archive.php → index.php
Category Archive → category-$slug.php → category-$id.php → category.php → archive.php
Custom Post Type Archive → archive-$posttype.php → archive.php
Custom Taxonomy Archive → taxonomy-$taxonomy-$term.php → taxonomy-$taxonomy.php → taxonomy.php → archive.php
Date Archive → Year Archive / Month Archive / Day Archive → date.php → archive.php
Tag Archive → tag-$slug.php → tag-$id.php → tag.php → archive.php

Singular Page
Single Post Page
Attachment Post → $mimetype-$subtype.php → $subtype.php → $mimetype.php → attachment.php → single.php → singular.php → index.php
Custom Post → If selected: $custom.php → single-$posttype-$slug.php → single-$posttype.php → single.php
Blog Post → If selected: $custom.php → single-post.php → single.php
Static Page → Page Template
Custom Template → $custom.php → page.php → singular.php
Default Template → page-$slug.php → page-$id.php → page.php

Site Front Page → front-page.php
Page Shown On Front → Page Template
Posts Shown On Front → home.php → index.php

Blog Posts Index page → home.php

Error 404 Page → 404.php → index.php

Search Result Page → search.php → index.php

For oEmbeds: embed-{post-type}-{post_format}.php → embed-{post-type}.php → embed.php → wp-includes/theme-compat/embed.php

🟪 Primary Template 🟩 Secondary Template 🟧 Variable Template ⬛ Page Type

Your First Plugin

Really, it's this easy

Good Documentation

http://codex.wordpress.org/Writing_a_Plugin

http://codex.wordpress.org/Plugin_API

Starting Off

First you create a subfolder inside of:

```
/wp-content/plugins
```

Call it whatever you'd like, such as `phpa-common`

```
/wp-content/plugins/phpa-common
```

Now inside of there, make a new file with the same name as the directory, so `phpa-common.php`

Basic Content

We will refer to this new file as your plugin file, inside of it you need to add in a block of comment that WordPress will parse to use as the description for UI:

```
<?php
/**
 * Plugin Name: php[architect] Common Code
 * Plugin URI: http://www.phparch.com/
 * Description: Provides various bits used on phparch.com
 * Version: 1.0
 * Author: Eli White
 * Author URI: http://eliw.com/
 * License: GPL2
 */
```

Officially: Done

That’s actually it. You’ve now created a plugin.

Granted, it doesn’t do anything yet. Go into your admin screen, go to the plugins tab, and you should see your new plugin.

Go ahead and enable it, even though it will do nothing.

Introducing the Hook System

How much of anything gets done in WordPress

Hook System

http://codex.wordpress.org/Plugin_API/Hooks

How most custom code gets activated in WordPress

Two categories of hooks: filters & actions

Implementing Filter Hooks

Filter hooks, allow you to change content on the fly.

There are thousands of different hooks for content:

http://codex.wordpress.org/Plugin_API/Filter_Reference

Example: Forcing title case rules onto your post titles:

```
function force_title_case($title, $id) {
    return ucwords($title);
}
add_filter('the_title', 'force_title_case', 10, 2);
```

Implementing Filter Hooks

Filter hooks, allow you to change content on the fly.

There are thousands of different hooks for content:

http://codex.wordpress.org/Plugin_API/Filter_Reference

Example: Forcing title case rules onto your post titles:

```
add_filter(‘the_title', function ($title, $id) {
    return ucwords($title);
}, 10, 2);
```

Action Hooks

Action hooks, set code to run at a specific point in the WP execution path

Over 600 different action hooks are defined:

http://codex.wordpress.org/Plugin_API/Action_Reference

```
function email_post($post_id) {
    if (!wp_is_post_revision($post_id)) return; // Don't send revisions
    $title = get_the_title($post_id);
    $content = get_the_content($post_id);
    $url = get_permalink($post_id);
    $subject = "Post Saved: {$title}";
    $message = "Updated:\n\n<a href=\"{$url}\">{$title}</a>\n\n{$content}";
    wp_mail('admin@example.com', $subject, $message);
}
add_action('save_post', 'email_post');
```

Let’s make a Shortcode

One of the simplest ways to add functionality

What’s a Shortcode?

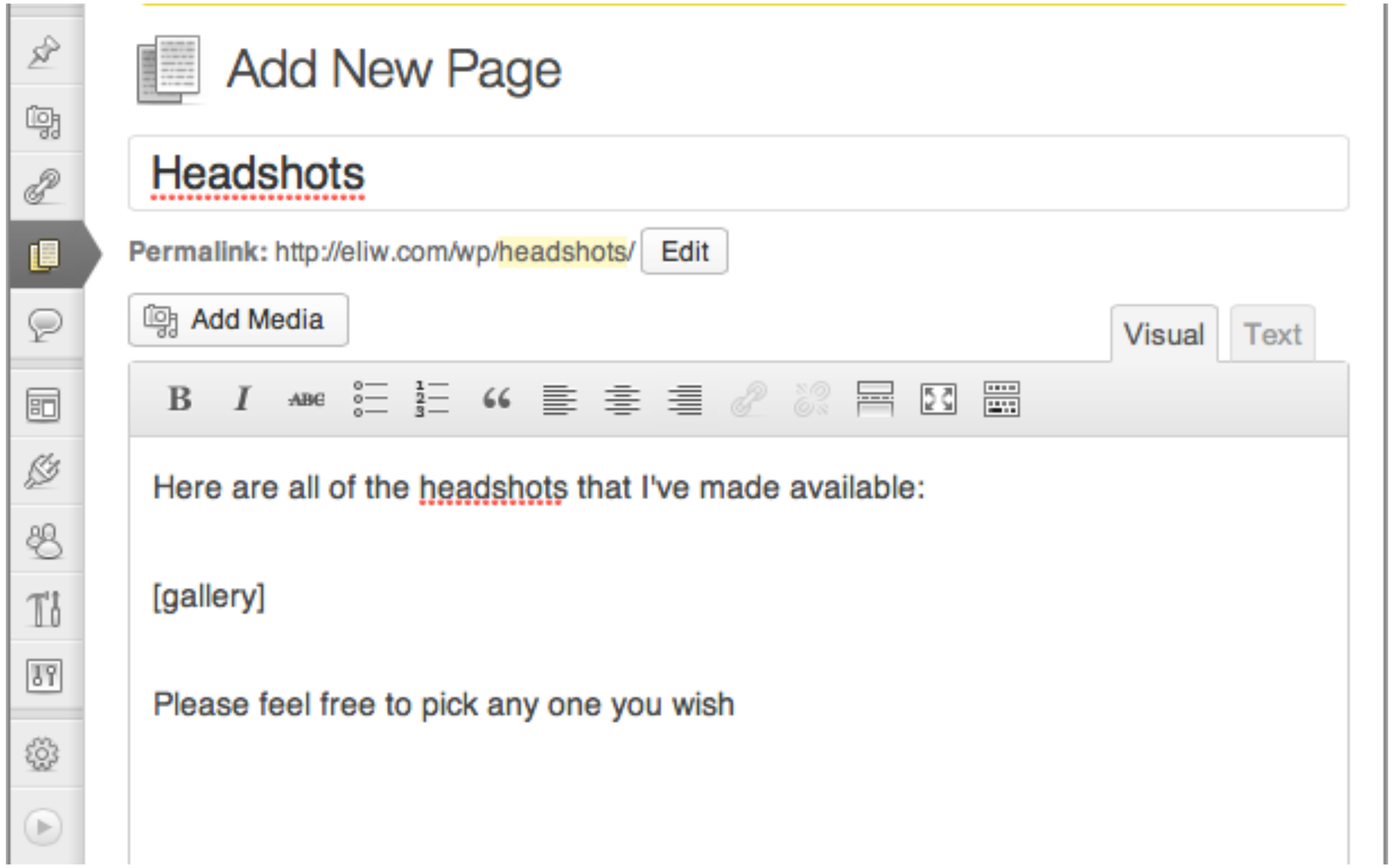
http://codex.wordpress.org/Shortcode_API

Shortcodes are a way of creating macros that are then used inside of post content.

Shortcode Format

Shortcodes are entered into a post as their name surrounded by brackets. For example, WordPress comes with the gallery shortcode creates a gallery of all attached media to a post:

```
[gallery]
```



Add New Page

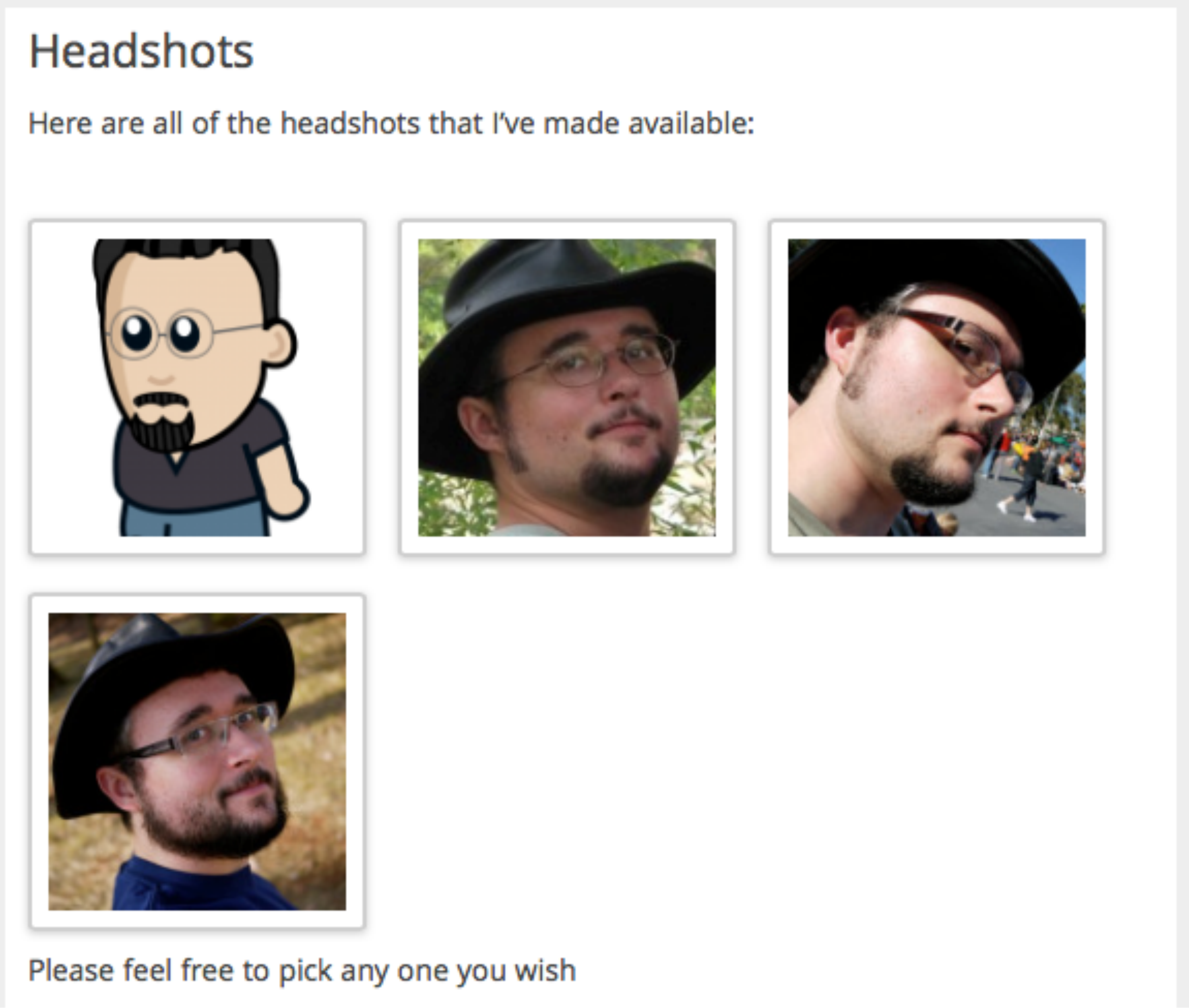
Headshots

Permalink: http://eliw.com/wp/headshots/

Here are all of the headshots that I've made available:

[gallery]

Please feel free to pick any one you wish



Headshots

Here are all of the headshots that I've made available:

Please feel free to pick any one you wish

Adding Parameters

The built in video shortcode, allows you to specify various formats for a video file you want to embed:

```
[video mp4="source.mp4" ogv="source.ogv" mov="source.mov" loop="on"]
```

It's also possible to have shortcodes that wrap content:

```
[caption width="200" caption="Rasmus Lerdorf"]
<img src="http://lerdorf.com/headshot.png" width="200" height="200" />
[/caption]
```

A Simple Shortcode

Shortcode to protect email from spam

```
function contact_us($attributes){
    $mailto = antispambot('mailto:contact@phparch.com');
    return '<a href="' . $mailto . '">Contact Us</a>';
}
add_shortcode('contact', 'contact_us');
```

Usage

```
[contact]
```

Handling Parameters

Parameters are passed into the `$attributes` field, but need additional processing via `shortcode_atts()`

```
function contact_us($attributes){
    extract(shortcode_atts([
        'email' => 'contact@phparch.com',
        'text' => 'Contact Us',
    ], $attributes));
    $mailto = antispambot("mailto:{$email}");
    return '<a href="' . $mailto . '">' . $text . '</a>';
}
add_shortcode('contact', 'contact_us');
```

Usage

```
[contact email="press@phparch.com" text="Press Department"]
```

Including Content

Make your tags wrap content, via accepting a `$content` parameter

```
function anti_mailto($attributes, $content = NULL){
    extract(shortcode_atts(array(
        'email' => 'contact@phparch.com',
    ), $attributes));
    $mailto = antispambot("mailto:{$email}");
    return '<a href="' . $mailto . '">' . $content . '</a>';
}
add_shortcode('antispam', 'anti_mailto');
```

Usage

```
[antispam email="write@phparch.com"]Write for Us![/antispam]
```

Nested Shortcodes

If you call `do_shortcode()` on the `$content` portion of your shortcode, you enable nesting

```
function anti_mailto($attributes, $content = NULL){
    extract(shortcode_atts(array(
        'email' => 'contact@phparch.com',
    ), $attributes));
    $mailto = antispambot("mailto:{$email}");
    return '<a href="' . $mailto . '">' .
           do_shortcode($content) . '</a>';
}
add_shortcode('antispam', 'anti_mailto');
```

Usage

```
[antispam email="write@phparch.com"]
    [rot13]Write for Us![/rot13]
[/antispam]
```

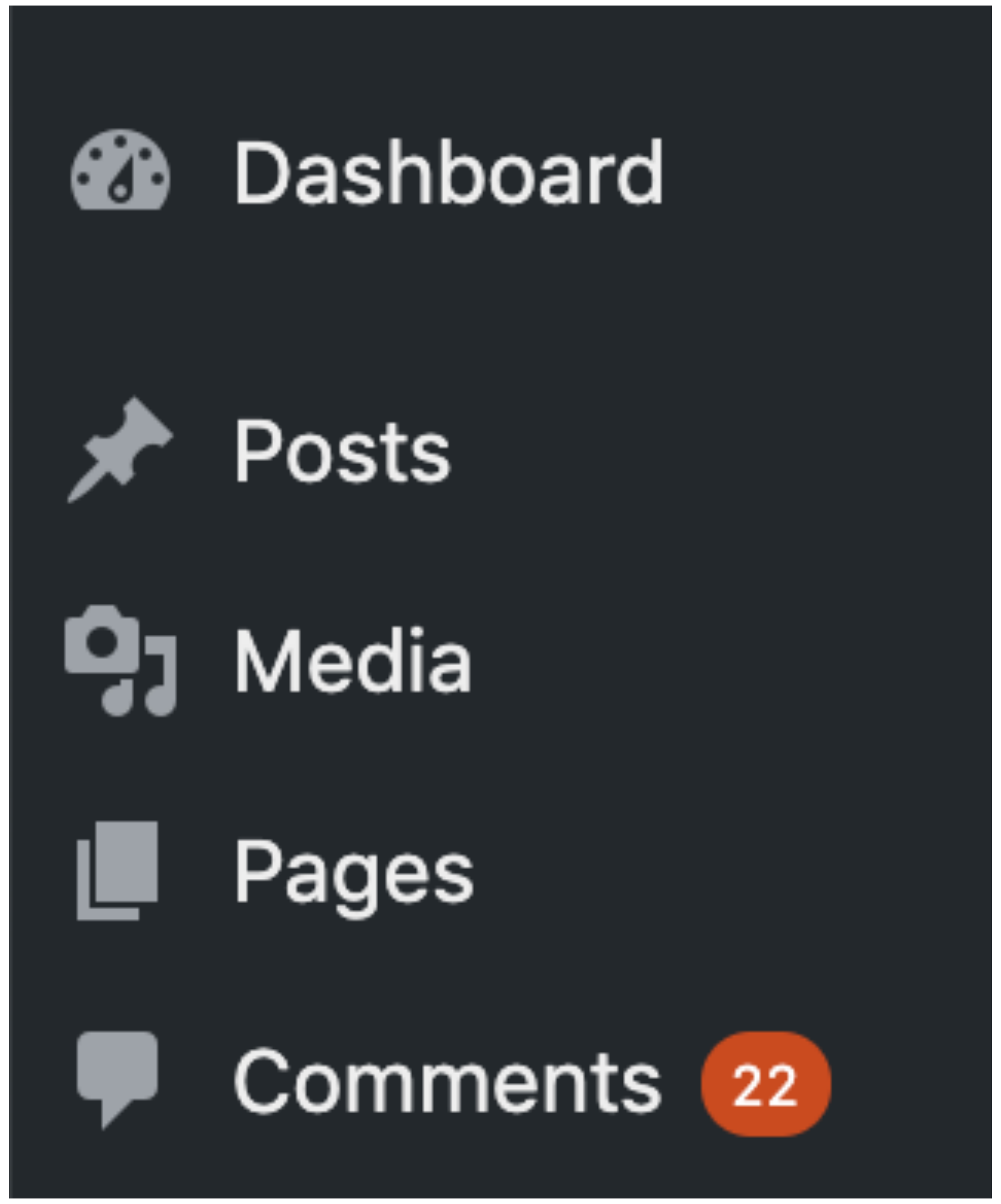
Custom Post Types

Very high level overview

Post Types?

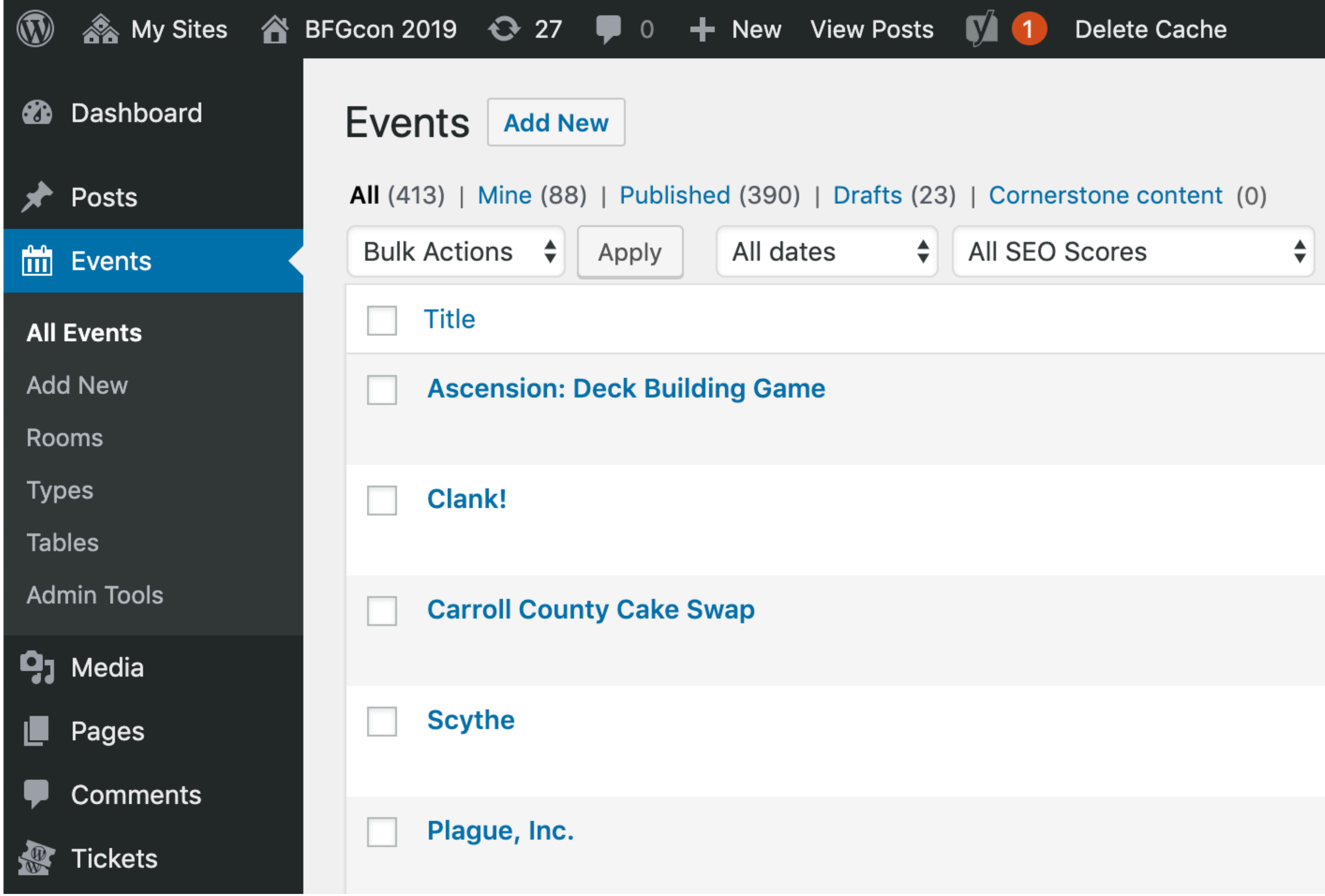
Posts are the basic 'storage unit' of WordPress

Posts
Media
Pages
Comments
Revisions
…more



Custom Types

Your own types can have any custom features you want.



Events

All (413) | Mine (88) | Published (390) | Drafts (23) | Cornerstone content (0)

□	Title
□	Ascension: Deck Building Game
□	Clank!
□	Carroll County Cake Swap
□	Scythe
□	Plague, Inc.

```
-outer.transparent header#top
nav .sf-menu > li.current_page
nav .sf-menu > li.current-menu-
nav > ul > li > a:hover > .sf-sub
nav ul #search-btn a:hover span,
nav .sf-menu > li.current-menu-i
hover .icon-salient-cart,.ascend
1!important;color:#ffffff!impo
rent header#top nav>ul>li.butt
widget-area-toggle a i.
-outer.transparent
```

Let’s look at the code!

```php
<?php
const ECS_ATTENDEE = 'ecs_event_attendee';
const ECS_GAMEMASTER = 'ecs_event_gamemaster';

/**
 * Register the 'Events' type that will be used for all events happening at the con:
 */
function ecs_events_init() {
  $labels = [
    'name'               => 'Events',
    'singular_name'      => 'Event',
    'menu_name'          => 'Events',
    'name_admin_bar'     => 'Event',
    'add_new'            => 'Add New',
    'add_new_item'       => 'Add New Event',
    'new_item'           => 'New Event',
    'edit_item'          => 'Edit Event',
    'view_item'          => 'View Event',
    'all_items'          => 'All Events',
    'search_items'       => 'Search Events',
    'parent_item_colon'  => 'Parent Events:',
    'not_found'          => 'No events found',
    'not_found_in_trash' => 'No events found in Trash',
      ];

  $args = [
    'labels'             => $labels,
    'description'        => 'Events that will be taking place at the Con',
    'public'             => true,
    'publicly_queryable' => true,
```

Custom Routing

Full blown custom framework type stuff

Intercepting Requests

Use ‘init’ action hook to stop WordPress theme system

Inject whatever code you wish

```php
<?php
// Grab specific URLs and don't let WP handle them
function intercept_request() {
    $path = parse_url($_SERVER['REQUEST_URI'], PHP_URL_PATH);
    switch ($path) {
        case '/speakers':
            speaker_page();
            exit;
        case '/spkr-headshot':
            $img = wp_get_attachment_image_src(
                 intval($_GET['id']), 'medium');
            wp_redirect($img[0]);
            exit;
    }
}
add_action("init", "intercept_request", 12);
```

What didn’t we cover?

…so so much

Gutenberg

Also…

- WP_Query & The Loop
- The entire Template system
  - Dynamic Sidebars
  - Custom Menus
  - Child Themes
- Post Types, Custom Post Types & Post Formats
- Other types of plugins:
  - Code Libraries
  - Drop-In Code Points
  - Drop-in Pages
- Plugin & Theme Options
- Modifying the Admin Pages
- Localization/I18n features
- Injecting data into JavaScript
- CSRF Protection
- Custom Endpoints / MVC
- Ajax in WordPress

… and much more

If you want to learn more…

http://codex.wordpress.org/

https://developer.wordpress.org/

Thank you very much!

But a brief commercial interruption

PHP[WORLD] 2019

Celebrating 25 years of PHP

Washington, D.C.

October 23-24

Be Inspired

Join the experts defining PHP's future for a 2 day conference. Expand your creativity while learning how others are solving problems.

Save 15%
WD9-PDX

world.phparch.com

Questions?

Twitter: @EliW

One for All Events:
www.oneforall.events

One for All Events

Bonus Content

… Did I really run that short?

Widgets

Magical Reusable Elements

What is a Widget?

Widgets are the ‘blocks’ of generated content that fill in the ‘Dynamic Sidebars’ of the theme.

While shortcodes are used in your content, Widgets are used in your design.

Basic Widget

```php
<?php
class Example_Widget extends WP_Widget {

    public function __construct() {
        // Needs to create the actual Widget within WordPress
    }

    public function widget( $args, $instance ) {
        // Will output the HTML/content of the Widget
    }

    public function form( $instance ) {
        // Creates the admin form, used to edit any configuration.
    }

    public function update( $new_instance, $old_instance ) {
        // Processes/Sanitizes any updates via the admin form.
    }
}
add_action( 'widgets_init', function(){
     register_widget( 'Example_Widget' );
});
```

Instantiating the Widget

To create the widget, you call the parent, passing in appropriate parameters:

```
public function __construct() {
     parent::__construct(
         'example_widget', // Base ID, must be unique
         'Example Widget', // The title/name of the Widget
         ['description' => 'A text widget built for the class']
         );
 }
```

At the moment really the only useful option that you can use is description.

Creating the Output

Inside of the widget method, you echo out the HTML that you wish to use. You are passed a number of default arguments that you should use to ensure a properly formatted widget:

```
public function widget( $args, $instance ) {
    echo $args['before_widget'];
    echo $args['before_title'], "Example Widget" , $args['after_title'];
    echo "<p>Built for this workshop!</p>”;
    echo $args['after_widget'];
}
```

Putting it all together

```php
<?php
class Example_Widget extends WP_Widget {

    public function __construct() {
        parent::__construct(
            'example_widget', // Base ID, must be unique
            'Example Widget', // The title/name of the Widget
            [ 'description' => 'A text widget built for the class' ]
            );
    }

    public function widget( $args, $instance ) {
        echo $args['before_widget'];
        echo $args['before_title'], "Example Widget" , $args['after_title'];
        echo "<p>Built for this workshop!</p>”;
        echo $args['after_widget'];
    }
}

add_action( 'widgets_init', function(){
     register_widget( 'Example_Widget' );
});
```

Making Widgets Configurable

You may have noticed that our widget didn't accept any kind of settings to configure what looks like.

It's possible to do this, we just need to implement the form & update methods on our object.

Creating the Form

Choose fields you want

Use helper methods to determine the appropriate ID and Name parameters

You receive current values of any fields as an array

```
public function form( $instance ) {
    $title = $instance['title'] ?? "Example Widget";
    $title_safe = esc_attr($title);
    $title_id = $this->get_field_id('title');
    $title_name = $this->get_field_name('title');
    echo <<<EOD
<p>
    <label>
        Title:
        <input id="{$title_id}" name=“{$title_name}"
               type="text" value="{$title_safe}" />
    </label>
</p>
EOD;
}
```

Accepting the Data

Next we can define the update method to handle any validation or sanitizing of the raw data.

```
public function update( $new, $old ) {
    $instance = [];
    $instance['title'] = $new['title'] ?? '';
    $instance['title'] = trim(strip_tags($instance['title']));
    return $instance;
}
```

(NOTE:You shouldn’t escape at this stage, only sanitize)

Using the Data

Now we can access this data inside of our widget display code. So we can update our display for example to be something like:

```php
public function widget( $args, $instance ) {
    echo $args['before_widget'];
    echo $args['before_title'], $instance['title'] , $args['after_title'];
    echo "<p>Built for this workshop!</p>”;
    echo $args['after_widget'];
}
```

Putting it all Together

```
class Example_Widget extends WP_Widget {
    public function __construct() {
        parent::__construct(‘example_widget', 'Example Widget’,
            [ 'description' => 'A text widget built for the class' ]);
    }

    public function widget( $args, $instance ) {
        echo $args[‘before_widget’], $args['before_title'], $instance[‘title’],
             $args[‘after_title’], "<p>Built for this workshop!</p>”, $args['after_widget'];
    }

    public function form( $instance ) {
        $title_safe = esc_attr(isset($instance[‘title’]) ?? ”Example Widget”);
        $title_id = $this->get_field_id('title');
        $title_name = $this->get_field_name('title');
        echo <<<EOD
<p><label>Title:
  <input id="{$title_id}" name=“{$title_name}” type="text" value="{$title_safe}" />
</label></p>
EOD;
    }

    public function update( $new, $old ) {
        $instance = [];
        $instance['title'] = trim(strip_tags($new['title'] ?? ‘’);
        return $instance;
    }
}
add_action( 'widgets_init', function(){ register_widget( 'Example_Widget' ); });
```