A Deep Dive
into the
Roles and Capabilities API

John Blackbourn

WordPress core developer
Working with WordPress for 12+ years
Senior Engineer at Human Made

@johnbillion

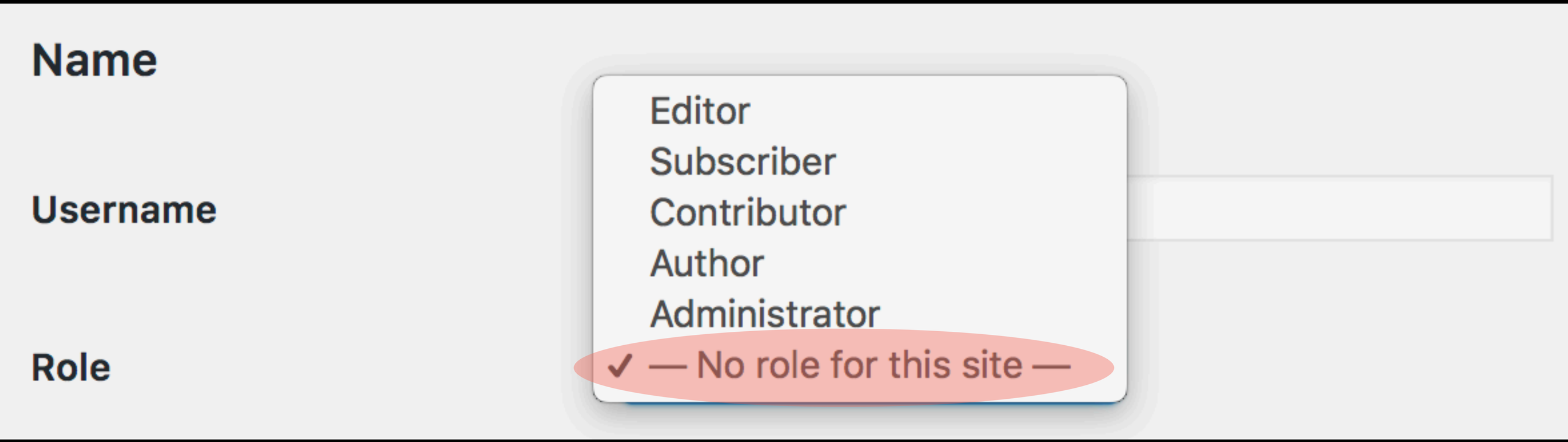
Roles and Caps

So I don't need to say the word *capabilities* correctly every time

Roles

- {Super Admin}
- Administrator
- Editor
- Author
- Contributor
- Subscriber
- {No role}

No Role


Name
Username
Role
Editor
Subscriber
Contributor
Author
Administrator
✓ — No role for this site —

Roles Aren't Actually Hierarchical

Capability	Super Admin	Administrator	Editor	Author	Contributor	Subscriber
moderate_comments	Y	Y	Y			
manage_categories	Y	Y	Y			
manage_links	Y	Y	Y			
edit_others_posts	Y	Y	Y			
edit_pages	Y	Y	Y			
edit_others_pages	Y	Y	Y			
edit_published_pages	Y	Y	Y			
publish_pages	Y	Y	Y			
delete_pages	Y	Y	Y			
delete_others_pages	Y	Y	Y			
delete_published_pages	Y	Y	Y			
delete_others_posts	Y	Y	Y			
delete_private_posts	Y	Y	Y			
edit_private_posts	Y	Y	Y			
read_private_posts	Y	Y	Y			
delete_private_pages	Y	Y	Y			
edit_private_pages	Y	Y	Y			
read_private_pages	Y	Y	Y			
unfiltered_html	Y	Y (single site)	Y (single site)			
Capability	Super Admin	Administrator	Editor	Author	Contributor	Subscriber
edit_published_posts	Y	Y	Y	Y		
upload_files	Y	Y	Y	Y		
publish_posts	Y	Y	Y	Y		
delete_published_posts	Y	Y	Y	Y		
edit_posts	Y	Y	Y	Y	Y	
delete_posts	Y	Y	Y	Y	Y	
Capability	Super Admin	Administrator	Editor	Author	Contributor	Subscriber
read	Y	Y	Y	Y	Y	Y

Roles, Capabilities, and Responsibilities

Role	Responsibility
• Administrator	*Administering the site*
• Editor	*Editing all content*
• Author	*Authoring posts*
• Contributor	*Contributing proposals*
• Subscriber	*Nothing*

Multiple Roles

- Administrator
- Editor
- Author
- Contributor
- Subscriber

} *Doesn't make much sense*

Multiple Roles

- Comment Moderator
- HR Manager

} *Two distinct responsibilities*

Roles and Responsibilities in bbPress

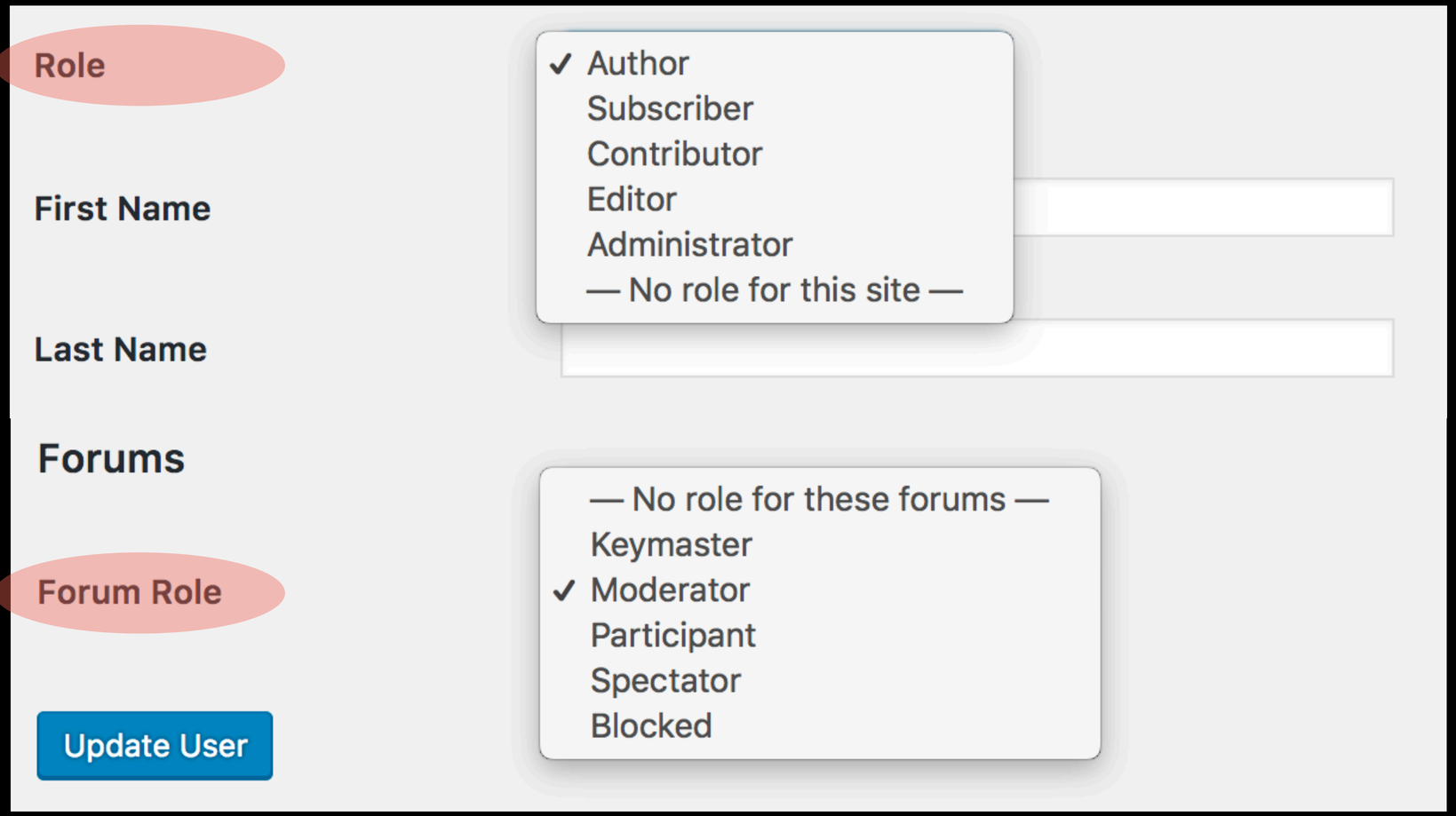
- Keymaster — *Managing the forums*
- Moderator — *Moderating discussion*
- Participant — *Participating in discussion*
- Spectator — *Following topics*

Multiple Roles in bbPress

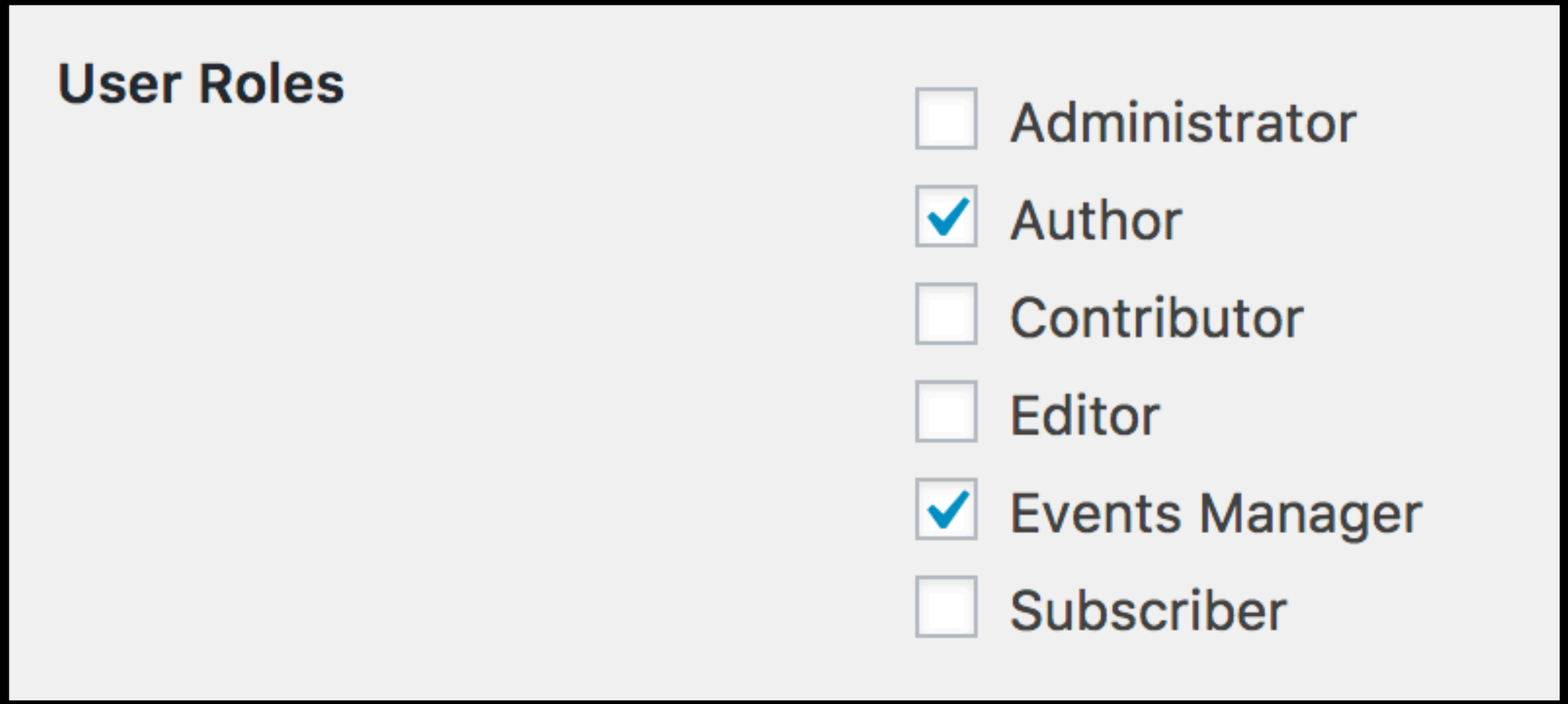
- Author
- Forum Moderator

} *Now we're talking*

Multiple Roles in bbPress



Multiple Roles with ‘Members’



Roles, Capabilities, and Responsibilities

Capability Checks

```
current_user_can( ‘edit_posts’ )
current_user_can( ‘manage_options’ )
current_user_can( ‘upload_files’ )

user_can( $user_id, ’delete_users’ )
```

Capability Checks

~~current_user_can('administrator')~~

```
current_user_can( 'create_users' )
```

Back In The Day

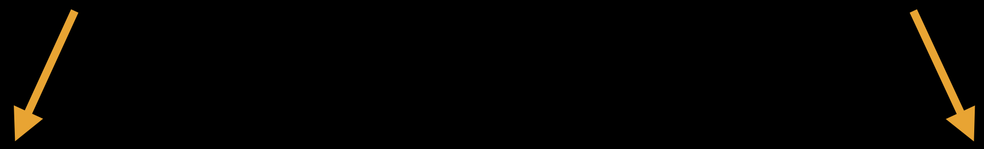
```
level_10
level_9
level_8
level_7
level_6
level_5
level_4
level_3
level_2
level_1
level_0
```

Meta Caps & Primitive Caps

```
current_user_can( ‘edit_post’, $post_id )
```

`map_meta_cap()`

Handles the mapping from
meta caps to primitive caps



`edit_post`

`edit_others_posts`

current_user_can(‘edit_post’, $post_id)

```
$post = get_post( $post_id );

if ( ! $post ) {
    // if the post doesn’t exist...
    $caps[] = ‘do_not_allow’;
    break;
}
```

current_user_can(‘edit_post’, $post_id)

```
if ( $user_id == $post->post_author ) {

  if ( $post->post_status == 'publish' ) {
    // If the post is published or scheduled...
    $caps[] = $post_type->cap->edit_published_posts;
  } else {
    // If the post is draft...
    $caps[] = $post_type->cap->edit_posts;
  }

}
```

current_user_can(‘edit_post’, $post_id)

```
} else {

  // The user is trying to edit someone else's post...
  $caps[] = $post_type->cap->edit_others_posts;

  // The post is published...
  if ( $post->post_status == 'publish' ) {
    $caps[] = $post_type->cap->edit_published_posts;
  } elseif ( 'private' == $post->post_status ) {
    $caps[] = $post_type->cap->edit_private_posts;
  }
```

Meta Caps & Primitive Caps

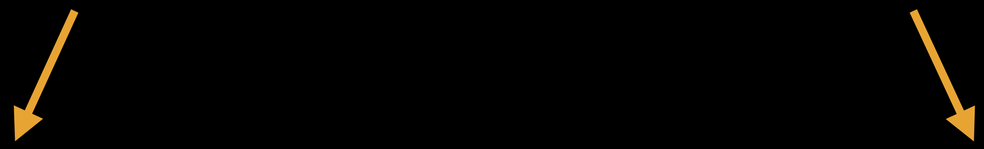
```
// another user's published post...
current_user_can( 'edit_post', $post_id )
    > edit_published_posts
    > edit_others_posts

// my own draft post
current_user_can( 'edit_post', $post_id )
    > edit_posts
```

current_user_can(‘delete_user’, $user_id)

```
case 'delete_user':
  // Only super admins can delete users.
  if ( is_multisite() && ! is_super_admin( $user_id ) )
    $caps[] = 'do_not_allow';
  else
    $caps[] = 'delete_users';
  break;
```

`map_meta_cap()`

Handles the mapping from
meta caps to primitive caps



`edit_post`

`edit_others_posts`

map_meta_cap

```
apply_filters( 'map_meta_cap', $caps, $cap, $user_id, $args );
```

```php
add_filter( 'map_meta_cap',
function( $required_caps, $cap, $user_id, $args ) {

    if ( 'delete_term' == $cap ) {
        // Prevent a "protected" term from being deleted:
        if ( get_term_meta( $args[0], 'protected', true ) ) {
            $required_caps[] = 'do_not_allow';
        }
    }

    return $required_caps;
}, 10, 4 );
```

```php
add_filter( 'map_meta_cap',
function( $required_caps, $cap, $user_id, $args ) {

    if ( 'publish_post' == $cap ) {
        // Introduce Lady Luck:
        if ( rand( 1, 6 ) !== 3 ) {
            $required_caps[] = 'do_not_allow';
        }
    }

    return $required_caps;
}, 10, 4 );
```

```php
add_filter( 'map_meta_cap',
function( $required_caps, $cap, $user_id, $args ) {

    if ( 'upload_files' == $cap ) {
        // If user can edit posts, allow to upload files:
        $required_caps = array( 'edit_posts' );
    }

    return $required_caps;
}, 10, 4 );
```

`map_meta_cap`

Use this filter to alter the required caps for an action at runtime

user_has_cap

```
apply_filters( 'user_has_cap', $caps, $required_caps, $args );
```

```
// Check authentication:
if ( ! current_user_can( 'switch_to_user', $user_id ) ) {
    wp_die( 'Could not switch users.' );
}
```

```php
add_filter( 'user_has_cap',
function( $user_caps, $required_caps, $args ) {
    if ( 'switch_to_user' === $args[0] ) {
        $user_caps['switch_to_user'] = (
            user_can( $args[1], 'edit_user', $args[2] ) &&
            $args[2] != $args[1]
        );
    }

    return $user_caps;
}, 10, 3 );
```

`map_meta_cap`

Use this filter to alter the required primitive caps for a cap check.

`user_has_cap`

Use this filter to grant or deny the actual caps of a user.

Recap

- Roles
- Responsibilities
- Capabilities
- `map_meta_cap`
- `user_has_cap`
- Trivia

Non-Logged-In Users

```
current_user_can( 'do', 'something' )

// This always returns true:
current_user_can( 'exist' )

// You need this too:
is_user_logged_in()
```

More Granular Caps in Core

```
current_user_can( 'edit_user', $user )
current_user_can( 'delete_user', $user )

// Coming soon:
current_user_can( 'edit_plugin', $plugin )
current_user_can( 'delete_plugin', $plugin )
current_user_can( 'activate_plugin', $plugin )
```

Core's Cap Tests

Tests_User_Capabilities

```
'upload_plugins'        => array( 'administrator' ),
'upload_themes'         => array( 'administrator' ),
'customize'             => array( 'administrator' ),
'add_users'             => array( 'administrator' ),

'edit_categories'       => array( 'administrator', 'editor' ),
'delete_categories'     => array( 'administrator', 'editor' ),
'manage_post_tags'      => array( 'administrator', 'editor' ),
'edit_post_tags'        => array( 'administrator', 'editor' ),
'delete_post_tags'      => array( 'administrator', 'editor' ),
'edit_css'              => array( 'administrator', 'editor' ),

'assign_categories'     => array( 'administrator', 'editor', 'author', 'contributor' ),
'assign_post_tags'      => array( 'administrator', 'editor', 'author', 'contributor' ),
```

John Blackbourn

@johnbillion

Questions?