```
<?php
/**
 * WordPress User Page
 *
 * Handles authentication, registering, resetting passwords, forgot
password,
 * and other user handling.
 *
 * @package WordPress
 */

/** Make sure that the WordPress bootstrap has run before
continuing. */
require( dirname( __FILE__ ) . '/wp-load.php' );

// Redirect to HTTPS login if forced to use SSL.
if ( force_ssl_admin() && ! is_ssl() ) {
        if ( 0 === strpos( $_SERVER['REQUEST_URI'], 'http' ) ) {

wp_safe_redirect( set_url_scheme( $_SERVER['REQUEST_URI'],
'https' ) );
                exit();
        } else {
                wp_safe_redirect( 'https://' .
$_SERVER['HTTP_HOST'] . $_SERVER['REQUEST_URI'] );
                exit();
        }
}

/**
 * Output the login page header.
 *
 * @since 2.1.0
 *
 * @param string   $title    Optional. WordPress login Page title to
display in the `<title>` element.
 *                           Default 'Log In'.
 * @param string   $message  Optional. Message to display in header.
Default empty.
 * @param WP_Error $wp_error Optional. The error to pass. Default is
a WP_Error instance.
 */
function login_header( $title = 'Log In', $message = '', $wp_error =
null ) {
        global $error, $interim_login, $action;

        // Don't index any of these forms
        add_action( 'login_head', 'wp_sensitive_page_meta' );

        add_action( 'login_head', 'wp_login_viewport_meta' );

        if ( ! is_wp_error( $wp_error ) ) {
                $wp_error = new WP_Error();
        }
```

```
        // Shake it!
        $shake_error_codes = array( 'empty_password',
'empty_email', 'invalid_email', 'invalidcombo', 'empty_username',
'invalid_username', 'incorrect_password' );
        /**
         * Filters the error codes array for shaking the login
form.
         *
         * @since 3.0.0
         *
         * @param array $shake_error_codes Error codes that shake
the login form.
         */
        $shake_error_codes = apply_filters( 'shake_error_codes',
$shake_error_codes );

        if ( $shake_error_codes && $wp_error->has_errors() &&
in_array( $wp_error->get_error_code(), $shake_error_codes ) ) {
                add_action( 'login_head', 'wp_shake_js', 12 );
        }

        $login_title = get_bloginfo( 'name', 'display' );

        /* translators: Login screen title. 1: Login screen name,
2: Network or site name */
        $login_title = sprintf( __( '%1$s &lsaquo; %2$s &#8212;
WordPress' ), $title, $login_title );

        if ( wp_is_recovery_mode() ) {
                /* translators: %s: Login screen title. */
                $login_title = sprintf( __( 'Recovery Mode &#8212;
%s' ), $login_title );
        }

        /**
         * Filters the title tag content for login page.
         *
         * @since 4.9.0
         *
         * @param string $login_title The page title, with extra
context added.
         * @param string $title       The original page title.
         */
        $login_title = apply_filters( 'login_title', $login_title,
$title );

        ?><!DOCTYPE html>
        <!--[if IE 8]>
                <html xmlns="http://www.w3.org/1999/xhtml"
class="ie8" <?php language_attributes(); ?>>
        <![endif]-->
        <!--[if !(IE 8) ]><!-->
                <html xmlns="http://www.w3.org/1999/xhtml" <?php
```

```
language_attributes(); ?>>
        <!--<![endif]-->
        <head>
        <meta http-equiv="Content-Type" content="<?php
bloginfo( 'html_type' ); ?>; charset=<?php bloginfo( 'charset' ); ?
>" />
        <title><?php echo $login_title; ?></title>
        <?php

        wp_enqueue_style( 'login' );

        /*
         * Remove all stored post data on logging out.
         * This could be added by add_action('login_head'...) like
wp_shake_js(),
         * but maybe better if it's not removable by plugins.
         */
        if ( 'loggedout' == $wp_error->get_error_code() ) {
                ?>
                <script>if("sessionStorage" in window){try{for(var
key in sessionStorage){if(key.indexOf("wp-autosave-")!=-1)
{sessionStorage.removeItem(key)}}}catch(e){}};</script>
                <?php
        }

        /**
         * Enqueue scripts and styles for the login page.
         *
         * @since 3.1.0
         */
        do_action( 'login_enqueue_scripts' );

        /**
         * Fires in the login page header after scripts are
enqueued.
         *
         * @since 2.1.0
         */
        do_action( 'login_head' );

        $login_header_url = __( 'https://wordpress.org/' );

        /**
         * Filters link URL of the header logo above login form.
         *
         * @since 2.1.0
         *
         * @param string $login_header_url Login header logo URL.
         */
        $login_header_url = apply_filters( 'login_headerurl',
$login_header_url );

        $login_header_title = '';
```

```
        /**
         * Filters the title attribute of the header logo above
login form.
         *
         * @since 2.1.0
         * @deprecated 5.2.0 Use login_headertext
         *
         * @param string $login_header_title Login header logo
title attribute.
         */
        $login_header_title = apply_filters_deprecated(
                'login_headertitle',
                array( $login_header_title ),
                '5.2.0',
                'login_headertext',
                __( 'Usage of the title attribute on the login logo
is not recommended for accessibility reasons. Use the link text
instead.' )
        );

        $login_header_text = empty( $login_header_title ) ?
__( 'Powered by WordPress' ) : $login_header_title;

        /**
         * Filters the link text of the header logo above the login
form.
         *
         * @since 5.2.0
         *
         * @param string $login_header_text The login header logo
link text.
         */
        $login_header_text = apply_filters( 'login_headertext',
$login_header_text );

        $classes = array( 'login-action-' . $action, 'wp-core-
ui' );
        if ( is_rtl() ) {
                $classes[] = 'rtl';
        }
        if ( $interim_login ) {
                $classes[] = 'interim-login';
                ?>
                <style type="text/css">html{background-color:
transparent;}</style>
                <?php

                if ( 'success' === $interim_login ) {
                        $classes[] = 'interim-login-success';
                }
        }
        $classes[] = ' locale-' .
sanitize_html_class( strtolower( str_replace( '_', '-',
get_locale() ) ) );
```

```
        /**
         * Filters the login page body classes.
         *
         * @since 3.5.0
         *
         * @param array  $classes An array of body classes.
         * @param string $action  The action that brought the
visitor to the login page.
         */
        $classes = apply_filters( 'login_body_class', $classes,
$action );

        ?>
        </head>
        <body class="login <?php echo esc_attr( implode( ' ',
$classes ) ); ?>">
        <?php
        /**
         * Fires in the login page header after the body tag is
opened.
         *
         * @since 4.6.0
         */
        do_action( 'login_header' );
        ?>
        <div id="login">
                <h1><a href="<?php echo
esc_url( $login_header_url ); ?>"><?php echo $login_header_text; ?
></a></h1>
        <?php
        /**
         * Filters the message to display above the login form.
         *
         * @since 2.1.0
         *
         * @param string $message Login message text.
         */
        $message = apply_filters( 'login_message', $message );
        if ( ! empty( $message ) ) {
                echo $message . "\n";
        }

        // In case a plugin uses $error rather than the $wp_errors
object.
        if ( ! empty( $error ) ) {
                $wp_error->add( 'error', $error );
                unset( $error );
        }

        if ( $wp_error->has_errors() ) {
                $errors   = '';
                $messages = '';
                foreach ( $wp_error->get_error_codes() as $code ) {
```

```
                        $severity = $wp_error-
>get_error_data( $code );
                        foreach ( $wp_error-
>get_error_messages( $code ) as $error_message ) {
                                if ( 'message' == $severity ) {
                                        $messages .= '    ' .
$error_message . "<br />\n";
                                } else {
                                        $errors .= '    ' .
$error_message . "<br />\n";
                                }
                        }
                }
                if ( ! empty( $errors ) ) {
                        /**
                         * Filters the error messages displayed
above the login form.
                         *
                         * @since 2.1.0
                         *
                         * @param string $errors Login error
message.
                         */
                        echo '<div id="login_error">' .
apply_filters( 'login_errors', $errors ) . "</div>\n";
                }
                if ( ! empty( $messages ) ) {
                        /**
                         * Filters instructional messages
displayed above the login form.
                         *
                         * @since 2.5.0
                         *
                         * @param string $messages Login messages.
                         */
                        echo '<p class="message">' .
apply_filters( 'login_messages', $messages ) . "</p>\n";
                }
        }
} // End of login_header()

/**
 * Outputs the footer for the login page.
 *
 * @since 3.1.0
 *
 * @param string $input_id Which input to auto-focus.
 */
function login_footer( $input_id = '' ) {
        global $interim_login;

        // Don't allow interim logins to navigate away from the
page.
        if ( ! $interim_login ) :
```

```
                ?>
        <p id="backtoblog"><a href="<?php echo
esc_url( home_url( '/' ) ); ?>">
                <?php
                /* translators: %s: site title */
                printf( _x( '&larr; Back to %s', 'site' ),
get_bloginfo( 'title', 'display' ) );
                ?>
        </a></p>
                <?php the_privacy_policy_link( '<div
class="privacy-policy-page-link">', '</div>' ); ?>
        <?php endif; ?>

        </div>

        <?php if ( ! empty( $input_id ) ) : ?>
        <script type="text/javascript">
        try{document.getElementById('<?php echo $input_id; ?
>').focus();}catch(e){}
        if(typeof wpOnload=='function')wpOnload();
        </script>
        <?php endif; ?>

        <?php
        /**
         * Fires in the login page footer.
         *
         * @since 3.1.0
         */
        do_action( 'login_footer' );
        ?>
        <div class="clear"></div>
        </body>
        </html>
        <?php
}

/**
 * Outputs the Javascript to handle the form shaking.
 *
 * @since 3.0.0
 */
function wp_shake_js() {
        ?>
<script type="text/javascript">
addLoadEvent = function(func){if(typeof jQuery!
="undefined")jQuery(document).ready(func);else if(typeof wpOnload!
='function'){wpOnload=func;}else{var
oldonload=wpOnload;wpOnload=function(){oldonload();func();}}};
function s(id,pos){g(id).left=pos+'px';}
function g(id){return document.getElementById(id).style;}
function shake(id,a,d){c=a.shift();s(id,c);if(a.length>0)
{setTimeout(function(){shake(id,a,d);},d);}
else{try{g(id).position='static';wp_attempt_focus();}catch(e){}}}
```

```
addLoadEvent(function(){ var p=new
Array(15,30,15,0,-15,-30,-15,0);p=p.concat(p.concat(p));var
i=document.forms[0].id;g(i).position='relative';shake(i,p,20);});
</script>
        <?php
}

/**
 * Outputs the viewport meta tag.
 *
 * @since 3.7.0
 */
function wp_login_viewport_meta() {
        ?>
        <meta name="viewport" content="width=device-width" />
        <?php
}

/**
 * Handles sending password retrieval email to user.
 *
 * @since 2.5.0
 *
 * @return bool|WP_Error True: when finish. WP_Error on error
 */
function retrieve_password() {
        $errors = new WP_Error();

        if ( empty( $_POST['user_login'] ) || !
is_string( $_POST['user_login'] ) ) {
                $errors->add( 'empty_username',
__( '<strong>ERROR</strong>: Enter a username or email
address.' ) );
        } elseif ( strpos( $_POST['user_login'], '@' ) ) {
                $user_data = get_user_by( 'email',
trim( wp_unslash( $_POST['user_login'] ) ) );
                if ( empty( $user_data ) ) {
                        $errors->add( 'invalid_email',
__( '<strong>ERROR</strong>: There is no account with that username
or email address.' ) );
                }
        } else {
                $login     = trim( $_POST['user_login'] );
                $user_data = get_user_by( 'login', $login );
        }

        /**
         * Fires before errors are returned from a password reset
request.
         *
         * @since 2.1.0
         * @since 4.4.0 Added the `$errors` parameter.
         *
         * @param WP_Error $errors A WP_Error object containing any
```

```
errors generated
         *                         by using invalid credentials.
         */
        do_action( 'lostpassword_post', $errors );

        if ( $errors->has_errors() ) {
                return $errors;
        }

        if ( ! $user_data ) {
                $errors->add( 'invalidcombo', __( '<strong>ERROR</
strong>: There is no account with that username or email
address.' ) );
                return $errors;
        }

        // Redefining user_login ensures we return the right case
in the email.
        $user_login = $user_data->user_login;
        $user_email = $user_data->user_email;
        $key        = get_password_reset_key( $user_data );

        if ( is_wp_error( $key ) ) {
                return $key;
        }

        if ( is_multisite() ) {
                $site_name = get_network()->site_name;
        } else {
                /*
                 * The blogname option is escaped with esc_html on
the way into the database
                 * in sanitize_option we want to reverse this for
the plain text arena of emails.
                 */
                $site_name =
wp_specialchars_decode( get_option( 'blogname' ), ENT_QUOTES );
        }

        $message = __( 'Someone has requested a password reset for
the following account:' ) . "\r\n\r\n";
        /* translators: %s: site name */
        $message .= sprintf( __( 'Site Name: %s' ), $site_name ) .
"\r\n\r\n";
        /* translators: %s: user login */
        $message .= sprintf( __( 'Username: %s' ), $user_login ) .
"\r\n\r\n";
        $message .= __( 'If this was a mistake, just ignore this
email and nothing will happen.' ) . "\r\n\r\n";
        $message .= __( 'To reset your password, visit the
following address:' ) . "\r\n\r\n";
        $message .= '<' . network_site_url( "wp-login.php?
action=rp&key=$key&login=" . rawurlencode( $user_login ),
'login' ) . ">\r\n";
```

```
        /* translators: Password reset notification email subject.
%s: Site title */
        $title = sprintf( __( '[%s] Password Reset' ),
$site_name );

        /**
         * Filters the subject of the password reset email.
         *
         * @since 2.8.0
         * @since 4.4.0 Added the `$user_login` and `$user_data`
parameters.
         *
         * @param string  $title      Default email title.
         * @param string  $user_login The username for the user.
         * @param WP_User $user_data  WP_User object.
         */
        $title = apply_filters( 'retrieve_password_title', $title,
$user_login, $user_data );

        /**
         * Filters the message body of the password reset mail.
         *
         * If the filtered message is empty, the password reset
email will not be sent.
         *
         * @since 2.8.0
         * @since 4.1.0 Added `$user_login` and `$user_data`
parameters.
         *
         * @param string  $message    Default mail message.
         * @param string  $key        The activation key.
         * @param string  $user_login The username for the user.
         * @param WP_User $user_data  WP_User object.
         */
        $message = apply_filters( 'retrieve_password_message',
$message, $key, $user_login, $user_data );

        if ( $message && ! wp_mail( $user_email,
wp_specialchars_decode( $title ), $message ) ) {
                wp_die( __( 'The email could not be sent. Possible
reason: your host may have disabled the mail() function.' ) );
        }

        return true;
}

//
// Main.
//

$action = isset( $_REQUEST['action'] ) ? $_REQUEST['action'] :
'login';
$errors = new WP_Error();
```

```
if ( isset( $_GET['key'] ) ) {
        $action = 'resetpass';
}

// Validate action so as to default to the login screen.
if ( ! in_array( $action, array( 'postpass', 'logout',
'lostpassword', 'retrievepassword', 'resetpass', 'rp', 'register',
'login', 'confirmaction',
WP_Recovery_Mode_Link_Service::LOGIN_ACTION_ENTERED ), true ) &&
false === has_filter( 'login_form_' . $action ) ) {
        $action = 'login';
}

nocache_headers();

header( 'Content-Type: ' . get_bloginfo( 'html_type' ) . ';
charset=' . get_bloginfo( 'charset' ) );

if ( defined( 'RELOCATE' ) && RELOCATE ) { // Move flag is set
        if ( isset( $_SERVER['PATH_INFO'] ) &&
( $_SERVER['PATH_INFO'] != $_SERVER['PHP_SELF'] ) ) {
                $_SERVER['PHP_SELF'] =
str_replace( $_SERVER['PATH_INFO'], '', $_SERVER['PHP_SELF'] );
        }

        $url = dirname( set_url_scheme( 'http://' .
$_SERVER['HTTP_HOST'] . $_SERVER['PHP_SELF'] ) );
        if ( $url != get_option( 'siteurl' ) ) {
                update_option( 'siteurl', $url );
        }
}

//Set a cookie now to see if they are supported by the browser.
$secure = ( 'https' === parse_url( wp_login_url(),
PHP_URL_SCHEME ) );
setcookie( TEST_COOKIE, 'WP Cookie check', 0, COOKIEPATH,
COOKIE_DOMAIN, $secure );
if ( SITECOOKIEPATH != COOKIEPATH ) {
        setcookie( TEST_COOKIE, 'WP Cookie check', 0,
SITECOOKIEPATH, COOKIE_DOMAIN, $secure );
}

/**
 * Fires when the login form is initialized.
 *
 * @since 3.2.0
 */
do_action( 'login_init' );

/**
 * Fires before a specified login form action.
 *
 * The dynamic portion of the hook name, `$action`, refers to the
```

```
action
 * that brought the visitor to the login form. Actions include
'postpass',
 * 'logout', 'lostpassword', etc.
 *
 * @since 2.8.0
 */
do_action( "login_form_{$action}" );

$http_post     = ( 'POST' == $_SERVER['REQUEST_METHOD'] );
$interim_login = isset( $_REQUEST['interim-login'] );

/**
 * Filters the separator used between login form navigation links.
 *
 * @since 4.9.0
 *
 * @param string $login_link_separator The separator used between
login form navigation links.
 */
$login_link_separator = apply_filters( 'login_link_separator', ' |
' );

switch ( $action ) {

        case 'postpass':
                if ( ! array_key_exists( 'post_password',
$_POST ) ) {
                        wp_safe_redirect( wp_get_referer() );
                        exit();
                }

                require_once ABSPATH . WPINC . '/class-phpass.php';
                $hasher = new PasswordHash( 8, true );

                /**
                 * Filters the life span of the post password
cookie.
                 *
                 * By default, the cookie expires 10 days from
creation. To turn this
                 * into a session cookie, return 0.
                 *
                 * @since 3.7.0
                 *
                 * @param int $expires The expiry time, as passed
to setcookie().
                 */
                $expire  = apply_filters( 'post_password_expires',
time() + 10 * DAY_IN_SECONDS );
                $referer = wp_get_referer();
                if ( $referer ) {
                        $secure = ( 'https' ===
parse_url( $referer, PHP_URL_SCHEME ) );
```

```
                } else {
                        $secure = false;
                }
                setcookie( 'wp-postpass_' . COOKIEHASH, $hasher-
>HashPassword( wp_unslash( $_POST['post_password'] ) ), $expire,
COOKIEPATH, COOKIE_DOMAIN, $secure );

                wp_safe_redirect( wp_get_referer() );
                exit();

        case 'logout':
                check_admin_referer( 'log-out' );

                $user = wp_get_current_user();

                wp_logout();

                if ( ! empty( $_REQUEST['redirect_to'] ) ) {
                        $redirect_to = $requested_redirect_to =
$_REQUEST['redirect_to'];
                } else {
                        $redirect_to           = 'wp-login.php?
loggedout=true';
                        $requested_redirect_to = '';
                }

                /**
                 * Filters the log out redirect URL.
                 *
                 * @since 4.2.0
                 *
                 * @param string  $redirect_to           The
redirect destination URL.
                 * @param string  $requested_redirect_to The
requested redirect destination URL passed as a parameter.
                 * @param WP_User $user                  The
WP_User object for the user that's logging out.
                 */
                $redirect_to = apply_filters( 'logout_redirect',
$redirect_to, $requested_redirect_to, $user );
                wp_safe_redirect( $redirect_to );
                exit();

        case 'lostpassword':
        case 'retrievepassword':
                if ( $http_post ) {
                        $errors = retrieve_password();
                        if ( ! is_wp_error( $errors ) ) {
                                $redirect_to = !
empty( $_REQUEST['redirect_to'] ) ? $_REQUEST['redirect_to'] : 'wp-
login.php?checkemail=confirm';
                                wp_safe_redirect( $redirect_to );
                                exit();
                        }
```

```
                }

                if ( isset( $_GET['error'] ) ) {
                        if ( 'invalidkey' == $_GET['error'] ) {
                                $errors->add( 'invalidkey',
__( 'Your password reset link appears to be invalid. Please request
a new link below.' ) );
                        } elseif ( 'expiredkey' ==
$_GET['error'] ) {
                                $errors->add( 'expiredkey',
__( 'Your password reset link has expired. Please request a new link
below.' ) );
                        }
                }

                $lostpassword_redirect = !
empty( $_REQUEST['redirect_to'] ) ? $_REQUEST['redirect_to'] : '';
                /**
                 * Filters the URL redirected to after submitting
the lostpassword/retrievepassword form.
                 *
                 * @since 3.0.0
                 *
                 * @param string $lostpassword_redirect The
redirect destination URL.
                 */
                $redirect_to =
apply_filters( 'lostpassword_redirect', $lostpassword_redirect );

                /**
                 * Fires before the lost password form.
                 *
                 * @since 1.5.1
                 * @since 5.1.0 Added the `$errors` parameter.
                 *
                 * @param WP_Error $errors A `WP_Error` object
containing any errors generated by using invalid
                 *                        credentials. Note that
the error object may not contain any errors.
                 */
                do_action( 'lost_password', $errors );

                login_header( __( 'Lost Password' ), '<p
class="message">' . __( 'Please enter your username or email
address. You will receive a link to create a new password via
email.' ) . '</p>', $errors );

                $user_login = '';

                if ( isset( $_POST['user_login'] ) &&
is_string( $_POST['user_login'] ) ) {
                        $user_login =
wp_unslash( $_POST['user_login'] );
                }
```

```
		?>

	<form name="lostpasswordform" id="lostpasswordform"
action="<?php echo esc_url( network_site_url( 'wp-login.php?
action=lostpassword', 'login_post' ) ); ?>" method="post">
	<p>
		<label for="user_login" ><?php _e( 'Username or
Email Address' ); ?><br />
		<input type="text" name="user_login"
id="user_login" class="input" value="<?php echo
esc_attr( $user_login ); ?>" size="20" autocapitalize="off" /></
label>
	</p>
		<?php
		/**
		 * Fires inside the lostpassword form tags, before
the hidden fields.
		 *
		 * @since 2.1.0
		 */
		do_action( 'lostpassword_form' );
		?>
		<input type="hidden" name="redirect_to" value="<?
php echo esc_attr( $redirect_to ); ?>" />
		<p class="submit"><input type="submit" name="wp-
submit" id="wp-submit" class="button button-primary button-large"
value="<?php esc_attr_e( 'Get New Password' ); ?>" /></p>
	</form>

	<p id="nav">
	<a href="<?php echo esc_url( wp_login_url() ); ?>"><?php
_e( 'Log in' ); ?></a>
		<?php
		if ( get_option( 'users_can_register' ) ) :
			$registration_url = sprintf( '<a
href="%s">%s</a>', esc_url( wp_registration_url() ),
__( 'Register' ) );

			echo esc_html( $login_link_separator );

			/** This filter is documented in wp-
includes/general-template.php */
			echo apply_filters( 'register',
$registration_url );
	endif;
		?>
	</p>

		<?php
		login_footer( 'user_login' );

		break;
```

```
        case 'resetpass':
        case 'rp':
                list( $rp_path ) = explode( '?',
wp_unslash( $_SERVER['REQUEST_URI'] ) );
                $rp_cookie       = 'wp-resetpass-' . COOKIEHASH;
                if ( isset( $_GET['key'] ) ) {
                        $value = sprintf( '%s:%s',
wp_unslash( $_GET['login'] ), wp_unslash( $_GET['key'] ) );
                        setcookie( $rp_cookie, $value, 0,
$rp_path, COOKIE_DOMAIN, is_ssl(), true );

wp_safe_redirect( remove_query_arg( array( 'key', 'login' ) ) );
                        exit;
                }

                if ( isset( $_COOKIE[ $rp_cookie ] ) && 0 <
strpos( $_COOKIE[ $rp_cookie ], ':' ) ) {
                        list( $rp_login, $rp_key ) = explode( ':',
wp_unslash( $_COOKIE[ $rp_cookie ] ), 2 );
                        $user                      =
check_password_reset_key( $rp_key, $rp_login );
                        if ( isset( $_POST['pass1'] ) && !
hash_equals( $rp_key, $_POST['rp_key'] ) ) {
                                $user = false;
                        }
                } else {
                        $user = false;
                }

                if ( ! $user || is_wp_error( $user ) ) {
                        setcookie( $rp_cookie, ' ', time() -
YEAR_IN_SECONDS, $rp_path, COOKIE_DOMAIN, is_ssl(), true );
                        if ( $user && $user->get_error_code() ===
'expired_key' ) {
                                wp_redirect( site_url( 'wp-
login.php?action=lostpassword&error=expiredkey' ) );
                        } else {
                                wp_redirect( site_url( 'wp-
login.php?action=lostpassword&error=invalidkey' ) );
                        }
                        exit;
                }

                $errors = new WP_Error();

                if ( isset( $_POST['pass1'] ) && $_POST['pass1'] !=
$_POST['pass2'] ) {
                        $errors->add( 'password_reset_mismatch',
__( 'The passwords do not match.' ) );
                }

                /**
                 * Fires before the password reset procedure is
validated.
```

```
                 *
                 * @since 3.5.0
                 *
                 * @param object           $errors WP Error object.
                 * @param WP_User|WP_Error $user   WP_User object
if the login and reset key match. WP_Error object otherwise.
                 */
                do_action( 'validate_password_reset', $errors,
$user );

                if ( ( ! $errors->has_errors() ) &&
isset( $_POST['pass1'] ) && ! empty( $_POST['pass1'] ) ) {
                        reset_password( $user, $_POST['pass1'] );
                        setcookie( $rp_cookie, ' ', time() -
YEAR_IN_SECONDS, $rp_path, COOKIE_DOMAIN, is_ssl(), true );
                        login_header( __( 'Password Reset' ), '<p
class="message reset-pass">' . __( 'Your password has been
reset.' ) . ' <a href="' . esc_url( wp_login_url() ) . '">' .
__( 'Log in' ) . '</a></p>' );
                        login_footer();
                        exit;
                }

                wp_enqueue_script( 'utils' );
                wp_enqueue_script( 'user-profile' );

                login_header( __( 'Reset Password' ), '<p
class="message reset-pass">' . __( 'Enter your new password
below.' ) . '</p>', $errors );

                ?>
        <form name="resetpassform" id="resetpassform" action="<?php
echo esc_url( network_site_url( 'wp-login.php?action=resetpass',
'login_post' ) ); ?>" method="post" autocomplete="off">
        <input type="hidden" id="user_login" value="<?php echo
esc_attr( $rp_login ); ?>" autocomplete="off" />

        <div class="user-pass1-wrap">
                <p>
                        <label for="pass1"><?php _e( 'New
password' ); ?></label>
                </p>

                <div class="wp-pwd">
                        <div class="password-input-wrapper">
                                <input type="password" data-
reveal="1" data-pw="<?php echo
esc_attr( wp_generate_password( 16 ) ); ?>" name="pass1" id="pass1"
class="input password-input" size="24" value="" autocomplete="off"
aria-describedby="pass-strength-result" />
                                <button type="button"
class="button button-secondary wp-hide-pw hide-if-no-js">
                                        <span class="dashicons
dashicons-hidden" aria-hidden="true"></span>
```

```
                                </button>
                        </div>
                        <div id="pass-strength-result"
class="hide-if-no-js" aria-live="polite"><?php _e( 'Strength
indicator' ); ?></div>
                </div>
                <div class="pw-weak">
                        <label>
                                <input type="checkbox"
name="pw_weak" class="pw-checkbox" />
                                <?php _e( 'Confirm use of weak
password' ); ?>
                        </label>
                </div>
        </div>

        <p class="user-pass2-wrap">
                <label for="pass2"><?php _e( 'Confirm new
password' ); ?></label><br />
                <input type="password" name="pass2" id="pass2"
class="input" size="20" value="" autocomplete="off" />
        </p>

        <p class="description indicator-hint"><?php echo
wp_get_password_hint(); ?></p>
        <br class="clear" />

                <?php
                /**
                 * Fires following the 'Strength indicator' meter
in the user password reset form.
                 *
                 * @since 3.9.0
                 *
                 * @param WP_User $user User object of the user
whose password is being reset.
                 */
                do_action( 'resetpass_form', $user );
                ?>
        <input type="hidden" name="rp_key" value="<?php echo
esc_attr( $rp_key ); ?>" />
        <p class="submit"><input type="submit" name="wp-submit"
id="wp-submit" class="button button-primary button-large" value="<?
php esc_attr_e( 'Reset Password' ); ?>" /></p>
        </form>

        <p id="nav">
        <a href="<?php echo esc_url( wp_login_url() ); ?>"><?php
_e( 'Log in' ); ?></a>
                <?php
                if ( get_option( 'users_can_register' ) ) :
                        $registration_url = sprintf( '<a
href="%s">%s</a>', esc_url( wp_registration_url() ),
__( 'Register' ) );
```

```
                        echo esc_html( $login_link_separator );

                        /** This filter is documented in wp-
includes/general-template.php */
                        echo apply_filters( 'register',
$registration_url );
        endif;
                ?>
        </p>

                <?php
                login_footer( 'user_pass' );

                break;

        case 'register':
                if ( is_multisite() ) {
                        /**
                         * Filters the Multisite sign up URL.
                         *
                         * @since 3.0.0
                         *
                         * @param string $sign_up_url The sign up
URL.
                         */

wp_redirect( apply_filters( 'wp_signup_location',
network_site_url( 'wp-signup.php' ) ) );
                        exit;
                }

                if ( ! get_option( 'users_can_register' ) ) {
                        wp_redirect( site_url( 'wp-login.php?
registration=disabled' ) );
                        exit();
                }

                $user_login = '';
                $user_email = '';

                if ( $http_post ) {
                        if ( isset( $_POST['user_login'] ) &&
is_string( $_POST['user_login'] ) ) {
                                $user_login =
$_POST['user_login'];
                        }

                        if ( isset( $_POST['user_email'] ) &&
is_string( $_POST['user_email'] ) ) {
                                $user_email =
wp_unslash( $_POST['user_email'] );
                        }
```

```
                        $errors = register_new_user( $user_login,
$user_email );
                        if ( ! is_wp_error( $errors ) ) {
                                $redirect_to = !
empty( $_POST['redirect_to'] ) ? $_POST['redirect_to'] : 'wp-
login.php?checkemail=registered';
                                wp_safe_redirect( $redirect_to );
                                exit();
                        }
                }

                $registration_redirect = !
empty( $_REQUEST['redirect_to'] ) ? $_REQUEST['redirect_to'] : '';
                /**
                 * Filters the registration redirect URL.
                 *
                 * @since 3.0.0
                 *
                 * @param string $registration_redirect The
redirect destination URL.
                 */
                $redirect_to =
apply_filters( 'registration_redirect', $registration_redirect );
                login_header( __( 'Registration Form' ), '<p
class="message register">' . __( 'Register For This Site' ) . '</
p>', $errors );
                ?>
        <form name="registerform" id="registerform" action="<?php
echo esc_url( site_url( 'wp-login.php?action=register',
'login_post' ) ); ?>" method="post" novalidate="novalidate">
        <p>
                <label for="user_login"><?php _e( 'Username' ); ?
><br />
                <input type="text" name="user_login"
id="user_login" class="input" value="<?php echo
esc_attr( wp_unslash( $user_login ) ); ?>" size="20"
autocapitalize="off" /></label>
        </p>
        <p>
                <label for="user_email"><?php _e( 'Email' ); ?
><br />
                <input type="email" name="user_email"
id="user_email" class="input" value="<?php echo
esc_attr( wp_unslash( $user_email ) ); ?>" size="25" /></label>
        </p>
                <?php
                /**
                 * Fires following the 'Email' field in the user
registration form.
                 *
                 * @since 2.1.0
                 */
                do_action( 'register_form' );
                ?>
```

```
        <p id="reg_passmail"><?php _e( 'Registration confirmation
will be emailed to you.' ); ?></p>
        <br class="clear" />
        <input type="hidden" name="redirect_to" value="<?php echo
esc_attr( $redirect_to ); ?>" />
        <p class="submit"><input type="submit" name="wp-submit"
id="wp-submit" class="button button-primary button-large" value="<?
php esc_attr_e( 'Register' ); ?>" /></p>
        </form>

        <p id="nav">
        <a href="<?php echo esc_url( wp_login_url() ); ?>"><?php
_e( 'Log in' ); ?></a>
                <?php echo esc_html( $login_link_separator ); ?>
        <a href="<?php echo esc_url( wp_lostpassword_url() ); ?
>"><?php _e( 'Lost your password?' ); ?></a>
        </p>

                <?php
                login_footer( 'user_login' );

                break;

        case 'confirmaction':
                if ( ! isset( $_GET['request_id'] ) ) {
                        wp_die( __( 'Missing request ID.' ) );
                }

                if ( ! isset( $_GET['confirm_key'] ) ) {
                        wp_die( __( 'Missing confirm key.' ) );
                }

                $request_id = (int) $_GET['request_id'];
                $key        =
sanitize_text_field( wp_unslash( $_GET['confirm_key'] ) );
                $result     =
wp_validate_user_request_key( $request_id, $key );

                if ( is_wp_error( $result ) ) {
                        wp_die( $result );
                }

                /**
                 * Fires an action hook when the account action has
been confirmed by the user.
                 *
                 * Using this you can assume the user has agreed to
perform the action by
                 * clicking on the link in the confirmation email.
                 *
                 * After firing this action hook the page will
redirect to wp-login a callback
                 * redirects or exits first.
                 *
```

```
                 * @since 4.9.6
                 *
                 * @param int $request_id Request ID.
                 */
                do_action( 'user_request_action_confirmed',
$request_id );

                $message =
_wp_privacy_account_request_confirmed_message( $request_id );

                login_header( __( 'User action confirmed.' ),
$message );
                login_footer();
                exit;

        case 'login':
        default:
                $secure_cookie   = '';
                $customize_login = isset( $_REQUEST['customize-
login'] );
                if ( $customize_login ) {
                        wp_enqueue_script( 'customize-base' );
                }

                // If the user wants SSL but the session is not
SSL, force a secure cookie.
                if ( ! empty( $_POST['log'] ) && !
force_ssl_admin() ) {
                        $user_name =
sanitize_user( $_POST['log'] );
                        $user      = get_user_by( 'login',
$user_name );

                        if ( ! $user && strpos( $user_name,
'@' ) ) {
                                $user = get_user_by( 'email',
$user_name );
                        }

                        if ( $user ) {
                                if ( get_user_option( 'use_ssl',
$user->ID ) ) {
                                        $secure_cookie = true;
                                        force_ssl_admin( true );
                                }
                        }
                }

                if ( isset( $_REQUEST['redirect_to'] ) ) {
                        $redirect_to = $_REQUEST['redirect_to'];
                        // Redirect to HTTPS if user wants SSL.
                        if ( $secure_cookie && false !==
strpos( $redirect_to, 'wp-admin' ) ) {
                                $redirect_to = preg_replace( '|
```

```
^http://|', 'https://', $redirect_to );
                        }
                } else {
                        $redirect_to = admin_url();
                }

                $reauth = empty( $_REQUEST['reauth'] ) ? false :
true;

                $user = wp_signon( array(), $secure_cookie );

                if ( empty( $_COOKIE[ LOGGED_IN_COOKIE ] ) ) {
                        if ( headers_sent() ) {
                                $user = new WP_Error(
                                        'test_cookie',
                                        sprintf(
                                                /* translators:
1: Browser cookie documentation URL, 2: Support forums URL */

__( '<strong>ERROR</strong>: Cookies are blocked due to unexpected
output. For help, please see <a href="%1$s">this documentation</a>
or try the <a href="%2$s">support forums</a>.' ),
                                                __( 'https://
wordpress.org/support/article/cookies/' ),
                                                __( 'https://
wordpress.org/support/' )
                                        )
                                );
                        } elseif ( isset( $_POST['testcookie'] )
&& empty( $_COOKIE[ TEST_COOKIE ] ) ) {
                                // If cookies are disabled we
can't log in even with a valid user+pass
                                $user = new WP_Error(
                                        'test_cookie',
                                        sprintf(
                                                /* translators:
%s: Browser cookie documentation URL */

__( '<strong>ERROR</strong>: Cookies are blocked or not supported by
your browser. You must <a href="%s">enable cookies</a> to use
WordPress.' ),
                                                __( 'https://
wordpress.org/support/article/cookies/#enable-cookies-in-your-
browser' )
                                        )
                                );
                        }
                }

                $requested_redirect_to =
isset( $_REQUEST['redirect_to'] ) ? $_REQUEST['redirect_to'] : '';
                /**
                 * Filters the login redirect URL.
                 *
```

```
                 * @since 3.0.0
                 *
                 * @param string           $redirect_to
The redirect destination URL.
                 * @param string           $requested_redirect_to
The requested redirect destination URL passed as a parameter.
                 * @param WP_User|WP_Error $user
WP_User object if login was successful, WP_Error object otherwise.
                 */
                $redirect_to = apply_filters( 'login_redirect',
$redirect_to, $requested_redirect_to, $user );

                if ( ! is_wp_error( $user ) && ! $reauth ) {
                        if ( $interim_login ) {
                                $message       = '<p
class="message">' . __( 'You have logged in successfully.' ) . '</
p>';
                                $interim_login = 'success';
                                login_header( '', $message );
                                ?>
                                </div>
                                <?php
                                /** This action is documented in
wp-login.php */
                                do_action( 'login_footer' );
                                ?>
                                <?php if ( $customize_login ) : ?>
                                <script type="text/
javascript">setTimeout( function(){ new
wp.customize.Messenger({ url: '<?php echo wp_customize_url(); ?>',
channel: 'login' }).send('login') }, 1000 );</script>
                        <?php endif; ?>
                                </body></html>
                                <?php
                                exit;
                        }

                        if ( ( empty( $redirect_to ) ||
$redirect_to == 'wp-admin/' || $redirect_to == admin_url() ) ) {
                                // If the user doesn't belong to a
blog, send them to user admin. If the user can't edit posts, send
them to their profile.
                                if ( is_multisite() && !
get_active_blog_for_user( $user->ID ) && ! is_super_admin( $user-
>ID ) ) {
                                        $redirect_to =
user_admin_url();
                                } elseif ( is_multisite() && !
$user->has_cap( 'read' ) ) {
                                        $redirect_to =
get_dashboard_url( $user->ID );
                                } elseif ( ! $user-
>has_cap( 'edit_posts' ) ) {
                                        $redirect_to = $user-
```

```
>has_cap( 'read' ) ? admin_url( 'profile.php' ) : home_url();
                                }

                                wp_redirect( $redirect_to );
                                exit();
                        }
                        wp_safe_redirect( $redirect_to );
                        exit();
                }

                $errors = $user;
                // Clear errors if loggedout is set.
                if ( ! empty( $_GET['loggedout'] ) || $reauth ) {
                        $errors = new WP_Error();
                }

                if ( empty( $_POST ) && $errors->get_error_codes()
=== array( 'empty_username', 'empty_password' ) ) {
                        $errors = new WP_Error( '', '' );
                }

                if ( $interim_login ) {
                        if ( ! $errors->has_errors() ) {
                                $errors->add( 'expired', __( 'Your
session has expired. Please log in to continue where you left
off.' ), 'message' );
                        }
                } else {
                        // Some parts of this script use the main
login form to display a message.
                        if ( isset( $_GET['loggedout'] ) && true
== $_GET['loggedout'] ) {
                                $errors->add( 'loggedout',
__( 'You are now logged out.' ), 'message' );
                        } elseif ( isset( $_GET['registration'] )
&& 'disabled' == $_GET['registration'] ) {
                                $errors->add( 'registerdisabled',
__( 'User registration is currently not allowed.' ) );
                        } elseif ( isset( $_GET['checkemail'] ) &&
'confirm' == $_GET['checkemail'] ) {
                                $errors->add( 'confirm',
__( 'Check your email for the confirmation link.' ), 'message' );
                        } elseif ( isset( $_GET['checkemail'] ) &&
'newpass' == $_GET['checkemail'] ) {
                                $errors->add( 'newpass',
__( 'Check your email for your new password.' ), 'message' );
                        } elseif ( isset( $_GET['checkemail'] ) &&
'registered' == $_GET['checkemail'] ) {
                                $errors->add( 'registered',
__( 'Registration complete. Please check your email.' ),
'message' );
                        } elseif ( strpos( $redirect_to,
'about.php?updated' ) ) {
                                $errors->add( 'updated',
```

```
__( '<strong>You have successfully updated WordPress!</strong>
Please log back in to see what&#8217;s new.' ), 'message' );
                        } elseif
( WP_Recovery_Mode_Link_Service::LOGIN_ACTION_ENTERED === $action )
{
                                $errors-
>add( 'enter_recovery_mode', __( 'Recovery Mode Initialized. Please
log in to continue.' ), 'message' );
                        }
                }

                /**
                 * Filters the login page errors.
                 *
                 * @since 3.6.0
                 *
                 * @param object $errors      WP Error object.
                 * @param string $redirect_to Redirect destination
URL.
                 */
                $errors = apply_filters( 'wp_login_errors',
$errors, $redirect_to );

                // Clear any stale cookies.
                if ( $reauth ) {
                        wp_clear_auth_cookie();
                }

                login_header( __( 'Log In' ), '', $errors );

                if ( isset( $_POST['log'] ) ) {
                        $user_login = ( 'incorrect_password' ==
$errors->get_error_code() || 'empty_password' == $errors-
>get_error_code() ) ? esc_attr( wp_unslash( $_POST['log'] ) ) : '';
                }
                $rememberme = ! empty( $_POST['rememberme'] );

                if ( $errors->has_errors() ) {
                        $aria_describedby_error = ' aria-
describedby="login_error"';
                } else {
                        $aria_describedby_error = '';
                }
                ?>

        <form name="loginform" id="loginform" action="<?php echo
esc_url( site_url( 'wp-login.php', 'login_post' ) ); ?>"
method="post">
        <p>
                <label for="user_login"><?php _e( 'Username or
Email Address' ); ?><br />
                <input type="text" name="log" id="user_login"<?php
echo $aria_describedby_error; ?> class="input" value="<?php echo
esc_attr( $user_login ); ?>" size="20" autocapitalize="off" /></
```

```
label>
        </p>
        <p>
                <label for="user_pass"><?php _e( 'Password' ); ?
><br />
                <input type="password" name="pwd" id="user_pass"<?
php echo $aria_describedby_error; ?> class="input" value=""
size="20" /></label>
        </p>
                <?php
                /**
                 * Fires following the 'Password' field in the
login form.
                 *
                 * @since 2.1.0
                 */
                do_action( 'login_form' );
                ?>
        <p class="forgetmenot"><label for="rememberme"><input
name="rememberme" type="checkbox" id="rememberme" value="forever" <?
php checked( $rememberme ); ?> /> <?php esc_html_e( 'Remember
Me' ); ?></label></p>
        <p class="submit">
                <input type="submit" name="wp-submit" id="wp-
submit" class="button button-primary button-large" value="<?php
esc_attr_e( 'Log In' ); ?>" />
                <?php   if ( $interim_login ) { ?>
                <input type="hidden" name="interim-login"
value="1" />
        <?php   } else { ?>
                <input type="hidden" name="redirect_to" value="<?
php echo esc_attr( $redirect_to ); ?>" />
        <?php } ?>
                <?php if ( $customize_login ) : ?>
                <input type="hidden" name="customize-login"
value="1" />
        <?php endif; ?>
                <input type="hidden" name="testcookie" value="1" />
        </p>
        </form>

                <?php if ( ! $interim_login ) { ?>
        <p id="nav">
                        <?php
                        if ( ! isset( $_GET['checkemail'] ) || !
in_array( $_GET['checkemail'], array( 'confirm', 'newpass' ) ) ) :
                                if
( get_option( 'users_can_register' ) ) :
                                        $registration_url =
sprintf( '<a href="%s">%s</a>', esc_url( wp_registration_url() ),
__( 'Register' ) );

                                        /** This filter is
documented in wp-includes/general-template.php */
```

```
                                        echo
apply_filters( 'register', $registration_url );

                                        echo
esc_html( $login_link_separator );
                                    endif;
                                    ?>
                <a href="<?php echo
esc_url( wp_lostpassword_url() ); ?>"><?php _e( 'Lost your
password?' ); ?></a>
                            <?php endif; ?>
        </p>
        <?php } ?>

        <script type="text/javascript">
        function wp_attempt_focus(){
        setTimeout( function(){ try{
                <?php if ( $user_login ) { ?>
        d = document.getElementById('user_pass');
        d.value = '';
        <?php } else { ?>
        d = document.getElementById('user_login');
                            <?php if ( 'invalid_username' == $errors-
>get_error_code() ) { ?>
        if( d.value != '' )
        d.value = '';
                                    <?php
                            }
        }
        ?>
        d.focus();
        d.select();
        } catch(e){}
        }, 200);
        }

                <?php
                /**
                 * Filters whether to print the call to
`wp_attempt_focus()` on the login screen.
                 *
                 * @since 4.8.0
                 *
                 * @param bool $print Whether to print the function
call. Default true.
                 */
                if ( apply_filters( 'enable_login_autofocus',
true ) && ! $error ) {
                            ?>
        wp_attempt_focus();
                <?php } ?>
        if(typeof wpOnload=='function')wpOnload();
                <?php if ( $interim_login ) { ?>
        (function(){
```

```
        try {
                var i, links = document.getElementsByTagName('a');
                for ( i in links ) {
                        if ( links[i].href )
                                links[i].target = '_blank';
                }
        } catch(e){}
        }());
        <?php } ?>
        </script>

                <?php
                login_footer();

                break;
} // End action switch.
```