

JavaScript Ajax

jQuery

- JavaScript
- jQuery
- AJAX
- PHP
- API
-

JavaScript

JavaScript ☐ WordPress ☐ CorePress ☒ WordPress ☐ jQuery ☐ jQuery ☐

jQuery

jQuery

WordPress jQuery jQuery HTML

```
jQuery(selector).event(function);
```

HTML

myplugin_settings.php

```
<form id="radioform">
<table>
<tbody>
<tr>
<td><input class="pref" checked="checked" name="book" type="radio" value="Sycamore Row" />Sycamore
Row</td>
<td>John Grisham</td>
</tr>
<tr>
<td><input class="pref" name="book" type="radio" value="Dark Witch" />Dark Witch</td>
<td>Nora Roberts</td>
</tr>
</tbody>
</table>
</form>
```

File not found or type unknown

AJAX jQuery usermeta: <script> PHP



```
$(".pref").change(function(){
    /*do stuff*/
});
```

“pref” “ ”

AJAX

AJAX

什么是 AJAX

AJAX 是 JavaScript 与 XML 的结合，用于在用户与 Internet 交互时，无需重新加载整个页面。

XML 是 PISON 的缩写。

AJAX 是 JavaScript 与 XML 的结合，用于在用户与 Internet 交互时，无需重新加载整个页面。

AJAX 是 JavaScript 与 XML 的结合，用于在用户与 Internet 交互时，无需重新加载整个页面。

什么是 AJAX

AJAX 是 JavaScript 与 XML 的结合，用于在用户与 Internet 交互时，无需重新加载整个页面。

AJAX 是 JavaScript 与 XML 的结合，用于在用户与 Internet 交互时，无需重新加载整个页面。

WordPress 使用 AJAX 来加载内容。WordPress 的 wp-load.php 文件位于 /wp-content/ 目录下。

AJAX 是 JavaScript 与 XML 的结合，用于在用户与 Internet 交互时，无需重新加载整个页面。

什么是 AJAX

WordPress 使用 AJAX 来加载内容。WordPress 的 wp-load.php 文件位于 /wp-content/ 目录下。

WordPress 使用 AJAX 来加载内容。WordPress 的 wp-load.php 文件位于 /wp-content/ 目录下。

1. 使用 JavaScript 与 jQuery 来发送 HTTP 请求。JavaScript 与 HTTP 请求。
2. 使用 JavaScript 与 jQuery 来发送 HTTP 请求。JavaScript 与 HTTP 请求。
3. 使用 AJAX 与 jQuery 来发送 HTTP 请求。JavaScript 与 HTTP 请求。

AJAX 与 jQuery

jQuery 的 \$.ajax() 方法用于发送 HTTP 请求。jQuery 的 \$.ajax() 方法用于发送 HTTP 请求。

```
var this2 = this;
```

WordPress AJAX wp-admin/admin-ajax.php . URL PHP jQuery jQuery URI
ajaxurl URL wp_localize_script(URI, jQuery PHP URL PHP [

my_ajax_obj.ajax_url

.ajaxurl()

```
{
  _ajax_nonce: my_ajax_obj.nonce, // nonce
  action: "my_tag_count", // action
  title: this.value // data
}
```

Nonce "Numberused ONCE" my_ajax_obj.nonce PHP URL jQuery

AJAX WordPress 24

Ajax

ajax_nonce PHP

_ajax_nonce: my_ajax_obj.nonce

WordPress AJAX action: "my_tag_count" AJAX

action: "my_tag_count"

XML JSON

jQuery this.value

title: this.value

XML JSON

```
function( data ) {
    this2.nextSibling.remove();
    $( this2 ).after( data );
}
```

var this2 = this; this2 ? this2 ? this2 ? /* do stuff */ this2

XML JSON

XML

XML AJAX "X" PHP PHP JSON 5.8 WordPress Internet Explorer

JSON

JSON eval() JSON PHP JSON PHP

PHP

jQuery Ajax

```
jQuery(document).ready(function($) { //wrapper
    $(".pref").change(function() { //event
```

```

var this2 = this;          //use in callback
$.post(my_ajax_obj.ajax_url, {    //POST request
  ajax_nonce: my_ajax_obj.nonce, //nonce
  action: "my_tag_count",        //action
  title: this.value              //data
}, function(data) {             //callback
  this2.nextSibling.remove(); //remove current title
  $(this2).after(data);        //insert server response
});
});
});
});

```

?????????????????? | /js/ myjquery.js

jQuery PHP Ajax

jQuery Ajax 使用 PHP 处理数据 jQuery 使用 jQuery PHP 使用 Ajax 使用

jQuery

WordPress 使用 Ajax 使用 `wp-admin/admin-ajax.php` 使用

jQuery

`wp_enqueue_script()` WordPress 使用

enqueue 使用

- **\$handle** 使用
- **\$src** `plugins_url()` 使用 URL - 使用
- **\$deps** 使用 'jquery' 使用 jQuery 使用 Ajax 使用
- **\$ver** 使用
- **\$args** `in_footer` `strategy` `defer` `async` `$in_footer` WordPress 6.3 使用/使用

```
wp_enqueue_script(
    'ajax-script',
    plugins_url( '/js/myjquery.js', __FILE__ ),
    array( 'jquery' ),
    '1.0.0',
    array(
        'in_footer' => true,
    )
);
```

`admin_enqueue_scripts` `wp_enqueue_scripts` `login_enqueue_scripts` 使用

`admin_enqueue_scripts` `is_home()` `is_single()` 使用

```
add_action( 'admin_enqueue_scripts', 'my_enqueue' );
function my_enqueue( $hook ) {
    if ( 'myplugin_settings.php' !== $hook ) {
        return;
    }
}
```

```

    }
    wp_enqueue_script(
        'ajax-script',
        plugins_url( '/js/myjquery.js', __FILE__ ),
        array( 'jquery' ),
        '1.0.0',
        array(
            'in_footer' => true,
        )
    );
}

```

WordPress 使用 jQuery 的 PHP 函数 `wp_enqueue_script()` 来注册和加载 jQuery 脚本。

WordPress

`wp_register_script()` 和 `wp_enqueue_script()` 是 WordPress 中用于管理脚本的函数。

WordPress

WordPress 使用 `wp_register_script()` 和 `wp_enqueue_script()` 来管理脚本。其中 `strategy` 参数用于指定脚本的加载策略。

WordPress

- `wp_enqueue_script()` 函数
 - `'strategy' => 'defer'` 表示脚本在 DOM 内容加载完成后，在 `DOMContentLoaded` 事件触发时加载。
 - `defer script` 表示脚本在 DOM 内容加载完成后，在 `DOMContentLoaded` 事件触发时加载。
- `wp_register_script()` 函数
 - `'strategy' => 'async'` 表示脚本在 DOM 内容加载过程中，异步加载。
 - `scri async` 表示脚本在 DOM 内容加载过程中，异步加载。

WordPress

```

wp_register_script(
    'ajax-script-two',
    plugins_url( '/js/myscript.js', __FILE__ ),
    array( 'jquery' ),
    '1.0.0',
    array(
        'strategy' => 'defer',
    )
);

```

wp_enqueue_script() 'ajax-script-two' 

\$args

--	--	--

jQuery AJAX PHP jQuery

```
$title_nonce = wp_create_nonce( 'title_example' );
```

title_example

--	--	--

jQuery PHP jQuery my_ajax_obj admin-ajax.php URL wp_localize_script()

```
wp_localize_script(
    'ajax-script',
    'my_ajax_obj',
    array(
        'ajax_url' => admin_url( 'admin-ajax.php' ),
        'nonce'    => $title_nonce,
    )
);
```

ajax-script

```
add_action( 'admin_enqueue_scripts', 'my_enqueue' );
```

```
/**
 * Enqueue my scripts and assets.
 *
 * @param $hook
 */
function my_enqueue( $hook ) {
    if ( 'myplugin_settings.php' !== $hook ) {
        return;
    }
}
```

```

wp_enqueue_script(
    'ajax-script',
    plugins_url( '/js/myjquery.js', __FILE__ ),
    array( 'jquery' ),
    '1.0.0',
    true
);

wp_localize_script(
    'ajax-script',
    'my_ajax_obj',
    array(
        'ajax_url' => admin_url( 'admin-ajax.php' ),
        'nonce'    => wp_create_nonce( 'title_example' ),
    )
);
}

```

```
current_user_can()
```

AJAX

PHP Ajax [JSON: via jQuery](#) WordPress

\$_GET **\$_POST** **\$_COOKIE** **\$_REQUEST**

\$_GET **\$_POST** **\$_COOKIE** **\$_REQUEST** **POST** **GET** **\$_REQUEST**

jQuery **action:** "my_tag_count" wp_ajax_my_tag_count . **wp_ajax_nopriv_my_tag_count**

```
add_action( 'wp_ajax_my_tag_count', 'my_ajax_handler' );
```

```

/**
 * Handles my AJAX request.
 */
function my_ajax_handler() {
    // Handle the ajax request here

```

```
wp_die(); // All ajax handlers die when finished
```

```
}

```

AJAX `check_ajax_referer()`

```
check_ajax_referer( 'title_example' );

```

```
wp_create_nonce()

```



```
$_POST['title'] . wp_unslash()

```

```
$title = wp_unslash( $_POST['title'] );

```

`update_user_meta()`

```
update_user_meta( get_current_user_id(), 'title_preference', sanitize_post_title( $title ) );

```



```
$args = array(
    'tag' => $title,
);
$query = new WP_Query( $args );

```

`jQuery`

XML

PHP XML `WP_Ajax_Response` `WP_Ajax_Response` XML >

JSON

`wp_send_json` `WP_Ajax_Response` `WordPress` `wp_send_json_success` `wp_send_json_error`
done() failed()



```
echo esc_html( $title ) . ' ( ' . $the_query->post_count . ' ) ';
```

jQuery



WP_Ajax_Response WP_send_json* wp_die() WordPress

AJAX

AJAX

```
/**
 * AJAX handler using JSON
 */
function my_ajax_handler_json() {
    check_ajax_referer( 'title_example' );
    $title = wp_unslash( $_POST['title'] );

    update_user_meta( get_current_user_id(), 'title_preference', sanitize_post_title( $title ) );

    $args = array(
        'tag' => $title,
    );
    $the_query = new WP_Query( $args );
    wp_send_json( esc_html( $title ) . ' ( ' . $the_query->post_count . ' ) ' );
}
```

```
/**
 * AJAX handler not using JSON.
 */
function my_ajax_handler() {
    check_ajax_referer( 'title_example' );
    $title = wp_unslash( $_POST['title'] );

    update_user_meta( get_current_user_id(), 'title_preference', sanitize_post_title( $title ) );
```

```
$args = array(
    'tag' => $title,
);
$query = new WP_Query( $args );
echo esc_html( $title ) . ' ( ' . $query->post_count . ' ) ';
wp_die(); // All ajax handlers should die when finished
}
```

Heartbeat API

Heartbeat API in WordPress `heartbeat.php` API

Overview

The heartbeat API is a REST API endpoint that allows a WordPress site to communicate with the WordPress core. It is used to send and receive data between the front end and the back end of the site.

The heartbeat API is used to send and receive data between the front end and the back end of the site.

1. `heartbeat-send` (Front end)
2. `heartbeat_received` (Back end)
3. `heartbeat-tick` (Back end)

The heartbeat API is used to send and receive data between the front end and the back end of the site.

API

The heartbeat API is a REST API endpoint that allows a WordPress site to communicate with the WordPress core. It is used to send and receive data between the front end and the back end of the site.

Usage

The heartbeat API is used to send and receive data between the front end and the back end of the site.

```
jQuery( document ).on( 'heartbeat-send', function ( event, data ) {  
    // Add additional data to Heartbeat data.  
    data.myplugin_customfield = 'some_data';  
});
```

Example

The heartbeat API is used to send and receive data between the front end and the back end of the site.

```
/**  
 * Receive Heartbeat data and respond.  
 *  
 * Processes data received via a Heartbeat request, and returns additional data to pass back to the front end.  
 */
```



```

* @param array $response Heartbeat response data to pass back to front end.
* @param array $data    Data received from the front end (unslashed).
*
* @return array
*/
function myplugin_receive_heartbeat( array $response, array $data ) {
    // If we didn't receive our data, don't send any back.
    if ( empty( $data['myplugin_customfield'] ) ) {
        return $response;
    }

    // Calculate our data and pass it back. For this example, we'll hash it.
    $received_data = $data['myplugin_customfield'];

    $response['myplugin_customfield_hashed'] = sha1( $received_data );
    return $response;
}

add_filter( 'heartbeat_received', 'myplugin_receive_heartbeat', 10, 2 );

```



```

jQuery( document ).on( 'heartbeat-tick', function ( event, data ) {
    // Check for our data, and use it.
    if ( ! data.myplugin_customfield_hashed ) {
        return;
    }

    alert( 'The hash is ' + data.myplugin_customfield_hashed );
});

```





WordPress uses jQuery and PHP

PHP

WordPress uses PHP

```
add_action( 'admin_enqueue_scripts', 'my_enqueue' );
function my_enqueue( $hook ) {
    if ( 'myplugin_settings.php' !== $hook ) {
        return;
    }

    wp_enqueue_script(
        'ajax-script',
        plugins_url( '/js/myjquery.js', __FILE__ ),
        array( 'jquery' ),
        '1.0.0',
        true
    );

    $title_nonce = wp_create_nonce( 'title_example' );
    wp_localize_script(
        'ajax-script',
        'my_ajax_obj',
        array(
            'ajax_url' => admin_url( 'admin-ajax.php' ),
            'nonce'    => $title_nonce,
        )
    );
}

add_action( 'wp_ajax_my_tag_count', 'my_ajax_handler' );
function my_ajax_handler() {
    check_ajax_referer( 'title_example' );
```

```

$title = wp_unslash( $_POST['title'] );

update_user_meta( get_current_user_id(), 'title_preference', $title );

$args = array(
    'tag' => $title,
);

$the_query = new WP_Query( $args );

echo esc_html( $title ) . ' ( ' . $the_query->post_count . ' ) ';

wp_die(); // all ajax handlers should die when finished
}

```

jQuery

js/myjquery.js

```

jQuery(document).ready(function($) {
    //wrapper
    $(".pref").change(function() {
        //event
        var this2 = this;
        //use in callback
        $.post(my_ajax_obj.ajax_url, {
            //POST request
            _ajax_nonce: my_ajax_obj.nonce, //nonce
            action: "my_tag_count",
            //action
            title: this.value
            //data
        }, function(data) {
            //callback
            this2.nextSibling.remove(); //remove the current title
            $(this2).after(data);
            //insert server response
        });
    });
});

```

- [WordPress AJAX](#)
- [WordPress Ajax](#)