区块注册

目录
使用 PHP 注册区块（服务器端）
通过元数据集合一次性注册所有区块（推荐方式）
单独注册元数据集合（ WordPress 6.7+）
使用 register_block_type() 注册单个区块
PHP-only blocks with auto-registration
Registering a block with JavaScript (client-side)
Additional resources

WordPress 中的区块通常以插件形式打包，并使用 block.json 元数据在服务器端和客户端进行注册。

虽然可以仅在客户端注册区块，但最佳实践强烈建议同时在服务器端和客户端进行注册。这种双重注册对于启
用服务器端功能至关重要，例如动态渲染、区块支持、区块钩子和样式变体。若缺少服务器端注册，这些功能
将无法正常运行。

例如，若希望区块 通过 theme.json 进行样式设置
，则必须在服务器端注册该区块。否则，区块将无法识别或应用 theme.json 中为其分配的任何样式。

下图详细说明了区块的注册流程。

打开区块注册流程图

使用 PHP 注册区块（服务器端）
在服务器端注册区块发生在主插件 PHP 文件中，并挂载到 init 钩子上。 WordPress
为此提供了多个函数，具体取决于你的需求以及目标 WordPress 的最低版本。

通过元数据集合一次性注册所有区块（推荐方

式）

自 WordPress 6.8 起，推荐的区块注册方式是使用 `wp_register_block_types_from_metadata_collection()`。此函数通过单次调用，注册 `blocks-manifest.php` 文件中定义的所有区块类型，从而无需再逐个调用 `register_block_type()`。

`blocks-manifest.php` 文件由 `wp-scripts build-blocks-manifest` 命令（或向 `build` 和 `start` 命令传递 `--blocks-manifest` 标志）自动生成。它会将项目中所有 `block.json` 文件的元数据编译到一个单独的 PHP 文件中。

```
function my_plugin_register_blocks() {
	wp_register_block_types_from_metadata_collection(
		__DIR__ . '/build',
		__DIR__ . '/build/blocks-manifest.php'
	);
}
add_action( 'init', 'my_plugin_register_blocks' );
```

这是 `create-block` 脚手架工具所采用的方法。它通过避免从磁盘读取多个 `block.json` 文件来提高性能，并简化了插件的注册代码——特别是对于包含多个区块的插件。

> `wp_register_block_types_from_metadata_collection()` 需要 WordPress 6.8 或更高版本。更多详情请参阅 WordPress 6.8 开发说明。

单独注册元数据集合（WordPress 6.7+）

如果你需要对单个区块的注册方式进行更多控制（例如，为每个区块传递不同的 `$args`），可以先使用 `wp_register_block_metadata_collection()` 注册清单，然后为每个区块单独调用 `register_block_type()`。WordPress 将从已加载的清单中读取每个区块的元数据，而不是从磁盘读取。

```
function my_plugin_register_blocks() {
	wp_register_block_metadata_collection(
		__DIR__ . '/build',
		__DIR__ . '/build/blocks-manifest.php'
	);

	register_block_type( __DIR__ . '/build/block-one' );
	register_block_type( __DIR__ . '/build/block-two', array(
		'render_callback' => 'my_plugin_render_block_two',
	) );
}
add_action( 'init', 'my_plugin_register_blocks' );
```

> `wp_register_block_metadata_collection()` 需要 WordPress 6.7 或更高版本。更多详情请参阅 WordPress 6.7 开发说明。

使用 `register_block_type()` 注册单个区块

`register_block_type()` 函数通过读取 `block.json` 文件中的元数据来注册单个区块类型。这仍然是一种有效的方法，特别是对于具有单个区块的插件或针对 WordPress 6.7 之前版本的情况。

此函数主要使用两个参数：

- **`$block_type` (`string`):** 包含 `block.json` 文件的目录路径，或者如果元数据文件名称不同，则为其完整路径。
- **`$args` (`array`):** 区块类型的可选附加参数数组。这可以包含一个 `render_callback`，作为 `block.json` 中 `render` 属性的替代方案。

在开发过程中，`block.json` 文件通常作为代码编译的一部分，从 `src` (源代码) 目录移动到 `build` 目录。因此，在注册区块时，请确保 `$block_type` 路径指向 `build` 目录内的 `block.json` 文件。

`register_block_type()` 函数在成功时返回注册的区块类型 (`WP_Block_Type`)，失败时返回 `false`。以下是一个使用 `render_callback` 的简单示例。

```
register_block_type(
	__DIR__ . '/build',
	array(
		'render_callback' => 'render_block_core_notice',
	)
);
```

以下是一个更完整的示例，包含了 `init` 钩子。

```
function minimal_block_ca6eda___register_block() {
	register_block_type( __DIR__ . '/build' );
}
add_action( 'init', 'minimal_block_ca6eda___register_block' );
```

查看 上面代码的 完整区块示例

PHP-only blocks with auto-registration

For blocks that only need server-side rendering, you can register them exclusively in PHP using the `autoRegister` flag and a `render_callback`. These blocks automatically appear in the editor without requiring any JavaScript registration or client-side code and use dynamic rendering.

```
register_block_type( 'my-plugin/server-block', array(
	'render_callback' => function( $attributes ) {
		$wrapper_attributes = get_block_wrapper_attributes();

		return sprintf(
			'<div %1$s>Server content</div>',
			$wrapper_attributes
		);
	},
	'supports' => array(
		'autoRegister' => true,
		'color' => array(
			'background' => true,
		),
	),
) );
```

Registering a block with JavaScript (client-side)

When the block has already been registered on the server and unless using PHP-only auto-registered blocks, you only need to register the client-side settings in JavaScript using the `registerBlockType` method from the `@wordpress/blocks` package. You just need to make sure you use the same block name as defined in the block's `block.json` file. Here's an example:

```
import { registerBlockType } from '@wordpress/blocks';

registerBlockType( 'my-plugin/notice', {
	edit: Edit,
	// ...other client-side settings
} );
```

While it's generally advised to register blocks on the server using PHP for the benefits outlined in the "Benefits using the metadata file" section, you can opt to register a block solely on the client-side. The `registerBlockType` method allows you to register a block type using metadata.

The function accepts two parameters:

- **`blockNameOrMetadata` (`string`|`Object`):** This can either be the block type's name as a string or an object containing the block's metadata, which is typically loaded from the `block.json` file.
- **`settings` (`Object`):** This is an object containing the block's client-side settings.

> You can import the contents of the `block.json` file (or any other `.json` file) directly into your JavaScript files if you're using a build process, such as the one provided by `wp-scripts`.

The `settings` object passed as the second parameter includes many properties, but these are the two most important ones:

- **`edit`:** The React component that gets used in the Editor for our block.
- **`save`:** The function that returns the static HTML markup that gets saved to the database.

The `registerBlockType()` function returns the registered block type (`WPBlock`) on success or `undefined` on failure. Here's an example:

```
import { registerBlockType } from '@wordpress/blocks';
import { useBlockProps } from '@wordpress/block-editor';
import metadata from './block.json';

const Edit = () => <p { ...useBlockProps() }>Hello World - Block Editor</p>;
const save = () => <p { ...useBlockProps.save() }>Hello World - Frontend</p>;

registerBlockType( metadata.name, {
	edit: Edit,
	save,
} );
```

See the full block example of the code above

Additional resources

- `wp_register_block_types_from_metadata_collection` PHP function (WordPress 6.8+)
- `wp_register_block_metadata_collection` PHP function (WordPress 6.7+)

- `register_block_type` PHP function
- `registerBlockType` JS function
- More efficient block type registration in 6.8 | WordPress Core dev note
- New block type registration APIs in 6.7 | WordPress Core dev note
- Why a block needs to be registered in both the server and the client? | GitHub Discussion
- Block Registration diagram

□□□ #2
□ □□□ □□□ 2026-03-26 17:08:43 UTC
□ □□□ □□□ 2026-03-26 17:08:43 UTC