Variations

- [□□□□□□ □□□□□]
- [□□□□□□ □□□□□]
- Registering block variations in PHP
- [□□□□□□ □□□□□]
- Block variations versus block styles
- [□□ isDefault]
  - [□□ isDefault □□□□□ □□□□□]
- [□□ isActive]
  - [isActive □□□□□□ □□□□□]

The Block Variations API allows you to define multiple versions (variations) of a block. A block variation differs from the original block by a set of initial attributes or inner blocks. When you insert the block variation into the Editor, these attributes and/or inner blocks are applied.

Variations are an excellent way to create iterations of existing blocks without building entirely new blocks from scratch.

To better understand this API, consider the Embed block. This block contains numerous variations for each type of embeddable content (WordPress, Youtube, etc.). Each Embed block variation shares the same underlying functionality for editing, saving, and so on. Besides the name and descriptive information, the main difference is the `providerNameSlug` attribute. Below is a simplified example of the variations in the Embed block. View the source code for the complete specification.

```
variations: [
□{
□□name: 'wordpress',
□□title: 'WordPress',
□□description: __( 'Embed a WordPress post.' ),
□□attributes: { providerNameSlug: 'wordpress' },
□},
□{
□□name: 'youtube',
```

```
□□title: 'YouTube',
□□description: __( 'Embed a YouTube video.' ),
□□attributes: { providerNameSlug: 'youtube' },
□},
],
```

定义区块变体

区块变体由一个可包含以下字段的对象定义：

- `name`（类型 `string`） – 唯一且机器可读的名称。
- `title`（可选，类型 `string`） – 人类可读的变体标题。
- `description`（可选，类型 `string`） – 人类可读的变体描述。
- `category`（可选，类型 `string`） - 用于在搜索界面中按类别排列区块类型的分类。
- `keywords`（可选，类型 `string[]`） - 帮助用户在搜索时发现该变体的术语数组（可翻译）。
- `icon`（可选，类型 `string` | `Object`） – 帮助可视化变体的图标。其形状可与区块类型相同。
- `attributes`（可选，类型 `Object`） – 覆盖区块属性的值。
- `innerBlocks`（可选，类型 `Array[]`） – 嵌套区块的初始配置。
- `example`（可选，类型 `Object`） – 为区块预览提供结构化数据。设置为 `undefined` 可禁用预览。详见 区块注册 API。
- `scope`（可选，类型 `WPBlockVariationScope[]`） - 默认为 `block` 和 `inserter`。变体适用的范围列表。可用选项包括：
  - `block` - 被区块用于筛选特定区块变体。 `Columns` 和 `Query` 区块具有此类变体，它们会传递给 实验性 BlockVariationPicker 组件。该组件负责显示变体并允许用户选择其中之一。
  - `inserter` - 区块变体显示在插入器中。
  - `transform` - 区块变体显示在变体转换组件中。
- `isDefault`（可选，类型 `boolean`） – 默认为 `false`。指示当前变体是否为默认变体（详见下文）。
- `isActive`（可选，类型 `Function|string[]`） - 一个函数或区块属性数组，用于确定选中区块时该变体是否处于活动状态。该函数接受 `blockAttributes` 和 `variationAttributes` 参数（详见下文）。

> 从技术上讲，您可以创建一个没有唯一 `name` 的区块变体，但 不推荐 这样做。唯一的 `name` 允许编辑器区分您的变体与其他可能存在的变体。它还允许在需要时注销您的变体，并且对 `isDefault` 设置有影响（详见下文）。

创建区块变体

□□□□□□□□□□□□□□□□ `variations`
□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□□ □□□□ API□

□□□□□□□□□□□□□□□□□□□□□ `wp.blocks.registerBlockVariation()`
□□□□□□□□□□□□□□□□□□□

```
wp.blocks.registerBlockVariation( 'core/embed', {
□name: 'custom-embed',
□attributes: { providerNameSlug: 'custom' },
} );
```

Registering block variations in PHP

Block variations can also be registered from PHP using the `get_block_type_variations` filter hook. This approach is particularly useful when you need to dynamically generate variations based on registered post types, taxonomies, or other WordPress data.

Here's an example of how to register a custom variation for the `core/image` block:

```
function my_custom_image_variation( $variations, $block_type ) {
    // Only modify variations for the image block
    if ( 'core/image' !== $block_type->name ) {
        return $variations;
    }

    // Add a custom variation
    $variations[] = array(
□□'name'        => 'wide-image',
□□'title'       => __( 'Wide image', 'textdomain' ),
□□'description' => __( 'A wide image', 'textdomain' ),
□□'scope'       => array( 'inserter' ),
□□'isDefault'   => false,
□□'attributes'  => array(
□□□'align' => 'wide', // Identifies the link type as custom
□□),
    );

    return $variations;
}
```

```
add_filter( 'get_block_type_variations', 'my_custom_image_variation', 10, 2 );
```

The `get_block_type_variations` filter is called when variations are requested for a block type. It receives two parameters:

- `$variations`: An array of currently registered variations for the block type
- `$block_type`: The full block type object

Note that variations registered through PHP will be merged with any variations registered through JavaScript using `registerBlockVariation()`.

> Check the How to register block variations with PHP blog post for more info about this

移除区块变体

区块变体同样可以轻松移除。为此，请使用 `wp.blocks.unregisterBlockVariation()` 函数。该函数接受区块名称和需要注销的变体 `name`。

```
wp.blocks.unregisterBlockVariation( 'core/embed', 'youtube' );
```

Block variations versus block styles

The main difference between block styles and block variations is that a block style just applies a CSS class to the block, so it can be styled in an alternative way. See the Block Styles API for more details.

If you want to apply initial attributes or inner blocks, this falls into block variation territory. It's also possible to override the default block style using the `className` attribute when defining a block variation.

```
variations: [
	{
		name: 'blue',
		title: __( 'Blue Quote' ),
		isDefault: true,
```

```
□□attributes: {
□□□color: 'blue',
□□□className: 'is-style-blue-quote'
□□},
□□icon: 'format-quote',
□□isActive: ( blockAttributes, variationAttributes ) =>
□□□blockAttributes.color === variationAttributes.color
□},
],
```

使用 `isDefault`

默认情况下，除了原始区块类型项外，所有变体都会显示在插入器中。但是，为列出的任何变体设置 `isDefault` 标志将覆盖插入器中的常规区块类型。这是一个根据特定需求定制编辑器体验的强大工具。

例如，如果你希望媒体与文本区块默认将图像显示在右侧，可以创建如下变体：

```
wp.blocks.registerBlockVariation( 'core/media-text', {
□name: 'media-text-media-right',
□title: __( 'Media & Text' ),
□isDefault: true,
□attributes: {
□□mediaPosition: 'right',
□},
} );
```

使用 `isDefault` 的注意事项

虽然 `isDefault` 在覆盖没有现有变体的区块时效果很好，但当存在其他变体时，您可能会遇到问题。

如果同一区块的另一个变体使用了 `isDefault`，您的变体不一定会成为默认值。编辑器会尊重第一个注册的带有 `isDefault` 的变体，这可能不是您的变体。

解决方案是在注册带有 `isDefault` 的变体之前，先注销其他变体。这一注意事项强化了始终为变体提供唯一 `name` 的建议。否则，该变体将无法被注销。

使用 `isActive`

虽然 `isActive` 属性是可选的，但建议使用。区块编辑器使用此 API
来检查哪个变体处于活动状态，并在编辑器中选择该变体的实例时显示正确的变体标题、图标和描述。

如果未设置 `isActive`，编辑器将无法区分原始区块实例和您的变体，因此将显示原始区块信息。

该属性可以设置为字符串数组（ `string[]`）或函数。建议尽可能使用字符串数组版本。

`string[]`
版本用于声明应将区块实例的哪些属性与给定变体的属性进行比较。将检查每个属性，如果所有属性都匹配，
则该变体将处于活动状态。

例如，在核心 Embed 区块中，`providerNameSlug` 属性用于确定嵌入提供程序（例如 'youtube' 或 'twitter'
）。变体可以这样声明：

```
const variations = [
□{
□□name: 'twitter',
□□title: 'Twitter',
□□icon: embedTwitterIcon,
□□keywords: [ 'tweet', __( 'social' ) ],
□□description: __( 'Embed a tweet.' ),
□□patterns: [ /^https?:\/\/(www\.)?twitter\.com\/.+/i ],
□□attributes: { providerNameSlug: 'twitter', responsive: true },
□},
□{
□□name: 'youtube',
□□title: 'YouTube',
□□icon: embedYouTubeIcon,
□□keywords: [ __( 'music' ), __( 'video' ) ],
□□description: __( 'Embed a YouTube video.' ),
□□patterns: [
□□□/^https?:\/\/((m|www)\.)?youtube\.com\/.+/i,
□□□/^https?:\/\/youtu\.be\/.+/i,
□□],
□□attributes: { providerNameSlug: 'youtube', responsive: true },
□},
□// ...
];
```

那么 `isActive` 属性将如下所示：

```
isActive: [ 'providerNameSlug' ];
```

这将导致 `providerNameSlug` 的区块实例值与变体声明中声明的值（上面代码片段中的值）进行比较，以确定哪个嵌入变体处于活动状态。

自 WordPress `6.6.0` 起，也支持嵌套对象路径。例如，考虑一个具有 `query` 对象作为属性的区块变体。可以仅基于该对象的 `postType` 属性（同时忽略其所有其他属性）来确定变体是否处于活动状态：

```
isActive: [ 'query.postType' ];
```

此属性的函数版本接受区块实例的 `blockAttributes` 作为第一个参数，以及为变体声明的 `variationAttributes` 作为第二个参数。可以通过比较这些参数并返回 `true` 或 `false`（指示此变体对此区块实例是否处于非活动状态）来确定变体是否处于活动状态。

使用相同的嵌入区块示例，函数版本将如下所示：

```
isActive: ( blockAttributes, variationAttributes ) =>
	blockAttributes.providerNameSlug === variationAttributes.providerNameSlug,
```

`isActive` 匹配的特异性

注意：自 *WordPress* `6.6.0` 版本起已改进处理方式。

如果存在多个变体，其 `isActive` 检查都匹配给定的区块实例，并且它们都是字符串数组，那么将选择具有最高特异性的变体。请考虑以下示例：

```
wp.blocks.registerBlockVariation( 'core/paragraph', {
	name: 'paragraph-red',
	title: 'Red Paragraph',
	attributes: {
		textColor: 'vivid-red',
	},
	isActive: [ 'textColor' ],
} );

wp.blocks.registerBlockVariation( 'core/paragraph', {
	name: 'paragraph-red-grey',
	title: 'Red/Grey Paragraph',
	attributes: {
		textColor: 'vivid-red',
		backgroundColor: 'cyan-bluish-gray',
```

```
},
isActive: [ 'textColor', 'backgroundColor' ],
} );
```

textColor: vivid-red backgroundColor: cyan-bluish-gray
isActive
isActive Red/Grey Paragraph

isActive string[]
isActive
string[] function

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