[illegible] API

[illegible]

- [illegible]
- [illegible]
- [illegible]
  - [illegible] 1[illegible]
  - Step 2: Add a button to the toolbar
  - Step 3: Apply a format when clicked
  - Step 4: Show the button only for specific blocks (Optional)
  - Step 5: Add a button outside of the dropdown (Optional)
- [illegible]
- [illegible]
- [illegible]

[illegible]

[illegible] API [illegible] _[illegible] _ [illegible]

[illegible] API [illegible]

[illegible] WordPress [illegible] _[illegible] _[illegible] [illegible] HTML [illegible] [illegible] `<samp>` HTML [illegible]

[illegible]

[illegible] WordPress [illegible] JavaScript [illegible] [illegible] [illegible] JavaScript [illegible] [illegible]

您需要准备：

- WordPress 开发环境
- 已激活并配置就绪的最小化插件
- 用于构建和入队的 JavaScript 设置

您可参考 完整的 format-api 示例 来配置您的环境。

分步指南

本指南将引用 `src/index.js` 作为进行修改的 JavaScript 文件。完成每个步骤后，运行 `npm run build` 会生成 `build/index.js`，该文件随后会加载到文章编辑器界面。

步骤 1：注册新格式

第一步是注册新格式，添加 `src/index.js` 文件并包含以下内容：

```
import { registerFormatType } from '@wordpress/rich-text';

registerFormatType( 'my-custom-format/sample-output', {
	title: 'Sample output',
	tagName: 'samp',
	className: null,
} );
```

可用格式类型的列表保存在 `core/rich-text` 存储中。您可以查询该存储以确认您的自定义格式现已可用。

在浏览器控制台中运行以下代码进行确认：

```
wp.data.select( 'core/rich-text' ).getFormatTypes();
```

它将返回一个包含格式类型的数组，其中也包括您自己的格式。

Step 2: Add a button to the toolbar

With the format available, the next step is to add a button to the UI by registering a component for the edit property.

Using the `RichTextToolbarButton` component, update `src/index.js`:

```
import { registerFormatType } from '@wordpress/rich-text';
import { RichTextToolbarButton } from '@wordpress/block-editor';

const MyCustomButton = ( props ) => {
	return (
		<RichTextToolbarButton
			icon="editor-code"
			title="Sample output"
			onClick={ () => {
				console.log( 'toggle format' );
			} }
		/>
	);
};

registerFormatType( 'my-custom-format/sample-output', {
	title: 'Sample output',
	tagName: 'samp',
	className: null,
	edit: MyCustomButton,
} );
```

Let's check that everything is working as expected. Build and reload and then select any block containing text like for example the paragraph block. Confirm the new button was added to the format toolbar.

Toolbar with custom button

Click the button and check the console.log for the "toggle format" message.

If you do not see the button or message, double check you are building and loading the JavaScript properly; and check the console.log for any errors.

Step 3: Apply a format when clicked

Next is to update the button to apply a format when clicked.

For our example, the `<samp>` tag format is binary - either a text selection has the tag or not, so we can use the `toggleFormat` method from the RichText package.

Update `src/index.js` changing the `onClick` action:

```
import { registerFormatType, toggleFormat } from '@wordpress/rich-text';
import { RichTextToolbarButton } from '@wordpress/block-editor';

const MyCustomButton = ( { isActive, onChange, value } ) => {
	return (
		<RichTextToolbarButton
			icon="editor-code"
			title="Sample output"
			onClick={ () => {
				onChange(
					toggleFormat( value, {
						type: 'my-custom-format/sample-output',
					} )
				);
			} }
			isActive={ isActive }
		/>
	);
};

registerFormatType( 'my-custom-format/sample-output', {
	title: 'Sample output',
	tagName: 'samp',
	className: null,
	edit: MyCustomButton,
} );
```

Confirm it is working: first build and reload, then make a text selection and click the button. Your browser will likely display that selection differently than the surrounding text.

You can also confirm by switching to HTML view (Code editor `Ctrl+Shift+Alt+M`) and see the text selection wrapped with `<samp>` HTML tags.

Use the `className` option when registering to add your own custom class to the tag. You can use that class and custom CSS to target that element and style as you wish.

Step 4: Show the button only for specific blocks (Optional)

By default, the button is rendered on every rich text toolbar (image captions, buttons, paragraphs, etc). You can render the button only on blocks of a certain type by using the data API.

Here is an example that only shows the button for Paragraph blocks:

```
import { registerFormatType, toggleFormat } from '@wordpress/rich-text';
import { RichTextToolbarButton } from '@wordpress/block-editor';
import { useSelect } from '@wordpress/data';

function ConditionalButton( { isActive, onChange, value } ) {
	const selectedBlock = useSelect( ( select ) => {
		return select( 'core/block-editor' ).getSelectedBlock();
	}, [] );

	if ( selectedBlock && selectedBlock.name !== 'core/paragraph' ) {
		return null;
	}

	return (
		<RichTextToolbarButton
			icon="editor-code"
			title="Sample output"
			onClick={ () => {
				onChange(
					toggleFormat( value, {
						type: 'my-custom-format/sample-output',
					} )
				);
			} }
			isActive={ isActive }
		/>
	);
}

registerFormatType( 'my-custom-format/sample-output', {
	title: 'Sample output',
	tagName: 'samp',
	className: null,
	edit: ConditionalButton,
} );
```

Step 5: Add a button outside of the dropdown (Optional)

Using the `RichTextToolbarButton` component, the button is added to the default dropdown menu. You can add the button directly to the toolbar by using the `BlockControls` component.

```
import { registerFormatType, toggleFormat } from '@wordpress/rich-text';
import { BlockControls } from '@wordpress/block-editor';
import { ToolbarGroup, ToolbarButton } from '@wordpress/components';

const MyCustomButton = ( { isActive, onChange, value } ) => {
	return (
		<BlockControls>
			<ToolbarGroup>
				<ToolbarButton
					icon="editor-code"
					title="Sample output"
					onClick={ () => {
						onChange(
							toggleFormat( value, {
								type: 'my-custom-format/sample-output',
							} )
						);
					} }
					isActive={ isActive }
				/>
			</ToolbarGroup>
		</BlockControls>
	);
};

registerFormatType( 'my-custom-format/sample-output', {
	title: 'Sample output',
	tagName: 'samp',
	className: null,
	edit: MyCustomButton,
} );
```

[illegible]

[illegible]

- [illegible] `npm run build`[illegible]
- [illegible]
- [illegible] JavaScript [illegible]
- [illegible]

[illegible]

[illegible]

- RichText[illegible] `registerFormatType`
- Components[illegible] `RichTextToolbarButton`
- RichText[illegible] `applyFormat`
- RichText[illegible] `removeFormat`
- RichText[illegible] `toggleFormat`

[illegible]

[illegible]
[illegible]

[illegible] block-development-examples [illegible] format-api [illegible]

[illegible] #2
[illegible] 2026-03-26 17:08:43 UTC
[illegible] 2026-03-26 17:08:43 UTC