@wordpress/plugins

- Installation
  - Plugins API
    - getPlugin
    - getPlugins
    - PluginArea
    - registerPlugin
    - unregisterPlugin
    - usePluginContext
    - withPluginContext
- Contributing to this package

Plugins module for WordPress.

Installation

Install the module

```
npm install @wordpress/plugins --save
```

*This package assumes that your code will run in an **ES2015+** environment. If you're using an environment that has limited or no support for such language features and APIs, you should include the polyfill shipped in `@wordpress/babel-preset-default` in your code.*

Plugins API

getPlugin

Returns a registered plugin settings.

Parameters

- *name* `string`: Plugin name.

Returns

- `WPPlugin | undefined`: Plugin setting.

getPlugins

Returns all registered plugins without a scope or for a given scope.

Parameters

- *scope* `string`: The scope to be used when rendering inside a plugin area. No scope by default.

Returns

- `WPPlugin[]`: The list of plugins without a scope or for a given scope.

PluginArea

A component that renders all plugin fills in a hidden div.

Usage

```
// Using ES5 syntax
var el = React.createElement;
var PluginArea = wp.plugins.PluginArea;

function Layout() {
    return el( 'div', { scope: 'my-page' }, 'Content of the page', PluginArea );
}
```

```
// Using ESNext syntax
import { PluginArea } from '@wordpress/plugins';

const Layout = () => (
    <div>
        Content of the page
        <PluginArea scope="my-page" />
    </div>
);
```

Parameters

- *props* `{ scope?: string; onError?: ( name: WPPlugin[ 'name' ], error: Error ) => void; }`:
- *props.scope* `string`:
- *props.onError* `( name: WPPlugin[ 'name' ], error: Error ) => void`:

Returns

- `Component`: The component to be rendered.

registerPlugin

Registers a plugin to the editor.

Usage

```
// Using ES5 syntax
var el = React.createElement;
var Fragment = wp.element.Fragment;
var PluginSidebar = wp.editor.PluginSidebar;
var PluginSidebarMoreMenuItem = wp.editor.PluginSidebarMoreMenuItem;
var registerPlugin = wp.plugins.registerPlugin;
var moreIcon = React.createElement( 'svg' ); //... svg element.

function Component() {
    return el(
        Fragment,
        {},
        el(
            PluginSidebarMoreMenuItem,
            {
                target: 'sidebar-name',
            },
            'My Sidebar'
        ),
        el(
            PluginSidebar,
            {
                name: 'sidebar-name',
                title: 'My Sidebar',
            },
            'Content of the sidebar'
        )
    );
```

```
}
registerPlugin( 'plugin-name', {
    icon: moreIcon,
    render: Component,
    scope: 'my-page',
} );
```

```
// Using ESNext syntax
import { PluginSidebar, PluginSidebarMoreMenuItem } from '@wordpress/editor';
import { registerPlugin } from '@wordpress/plugins';
import { more } from '@wordpress/icons';

const Component = () => (
    <>
        <PluginSidebarMoreMenuItem target="sidebar-name">
            My Sidebar
        </PluginSidebarMoreMenuItem>
        <PluginSidebar name="sidebar-name" title="My Sidebar">
            Content of the sidebar
        </PluginSidebar>
    </>
);

registerPlugin( 'plugin-name', {
    icon: more,
    render: Component,
    scope: 'my-page',
} );
```

Parameters

- *name* `string`: A string identifying the plugin. Must be unique across all registered plugins.
- *settings* `PluginSettings`: The settings for this plugin.

Returns

- `PluginSettings | null`: The final plugin settings object.

unregisterPlugin

Unregisters a plugin by name.

Usage

```
// Using ES5 syntax
var unregisterPlugin = wp.plugins.unregisterPlugin;

unregisterPlugin( 'plugin-name' );
```

```
// Using ESNext syntax
import { unregisterPlugin } from '@wordpress/plugins';

unregisterPlugin( 'plugin-name' );
```

Parameters

- *name* `string`: Plugin name.

Returns

- `WPPlugin | undefined`: The previous plugin settings object, if it has been successfully unregistered; otherwise `undefined`.

usePluginContext

A hook that returns the plugin context.

Returns

- `PluginContext`: Plugin context

withPluginContext

> **Deprecated** 6.8.0 Use `usePluginContext` hook instead.

A Higher Order Component used to inject Plugin context to the wrapped component.

Parameters

- *mapContextToProps* `( context: PluginContext, props: T ) => T & PluginContext`: Function called on every context change, expected to return object of props to merge with the component's own props.

Returns

- `Component`: Enhanced component with injected context as props.

Contributing to this package

This is an individual package that’s part of the Gutenberg project. The project is organized as a monorepo. It’s made up of multiple self-contained software packages, each with a specific purpose.

The packages in this monorepo are published to npm and used by WordPress as well as other software projects.

To find out more about contributing to this package or Gutenberg as a whole, please read the project’s main contributor guide.

□□□ #2
□ □□□ □□□ 2026-10-31 18:19:52 UTC
□ □□□ □□□ 2026-10-31 18:19:54 UTC