@wordpress/edit-widgets

- Batch processing
- Installation
- How this works
- Contributing to this package

Widgets Page Module for WordPress.

> This package is meant to be used only with WordPress core. Feel free to use it in your own project but please keep in mind that it might never get fully documented.

Batch processing

This package contains the first version of what may eventually become `@wordpress/batch-processing` package. Once imported, `core/__experimental-batch-processing` store gets registered. As the name says – it is highly experimental and considered a private API for now.

Installation

Install the module

```
npm install @wordpress/edit-widgets
```

*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.*

How this works

The new Widgets screen in WordPress admin is another block editor, just like the Post editor or the experimental site editor. Hence it will be referred often as the Widgets editor.

This editor manages widget areas and offers a way to add Gutenberg blocks to them, in addition to regular widgets. To support both widgets and blocks, the editor employs a translation mechanism between widget storage and block grammar.

There is a widget block that acts as a block UI for the widget data. This block is instantiated by default with a list of all available widgets to choose from. The block wraps its functionality in two modes: edit and preview based on the selected widget. The widget block's edit mode shows the standard Widget form, while the preview does a server-side render of the widget.

There is a block widget that acts as a storage mechanism for blocks added to widget areas. This widget is a special case of the HTML widget, where the block data is stored as it is rendered by the block's save function. All blocks added to widget areas are stored as these special HTML widgets, in one type of widget, the block widget.

This mechanism, using a widget block to edit widgets as blocks and a block widget to store blocks as widgets, ensures 100% compatibility with the old Widgets screen. Thus, if the new Widget editor, which is block-based, breaks some widgets' functionality that depends on the admin page's HTML structure or jQuery events, it is easy to revert to the old screen and continue to edit the legacy widgets.

Being just a block editor, the Widgets editor needs REST API entity management endpoints. For support, two new endpoints have been added: ./widgets and /sidebars. The ./widgets endpoint is used to load and save widgets and retrieve a server-side render of the widget's edit form. The /sidebars endpoint is used to list widget areas and assign or remove a widget to or from a widget area. There is also an /widget-types endpoint listing what kind widgets are available, e.g. text widget, calendar widget etc

In order to make the experience as seamless as possible for users, the following "magic" happens in the Widgets editor:

- for every available widget, a variation of the widget block is registered so that the user can see and search by the exact name of what they need
- all widgets that have a block equivalent (a block that fulfills the same function) can be made not available as a widget block variation via a filter
- all core widgets that have a block equivalent are not available as a widget block variation

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 18:18:49 UTC
□ □□□ □□□ 2026-10-31 18:18:50 UTC