@wordpress/interface

- Installation
- API Usage
  - Complementary Areas
  - Pinned Items
  - Preferences
    - Features
- Contributing to this package

> This package is still experimental. “Experimental” means this is an early implementation subject to drastic and breaking changes.

The Interface Package contains the basis to start a new WordPress screen as Edit Post or Edit Site. The package offers a data store and a set of components. The store is useful to contain common data required by a screen (e.g., active areas). The information is persisted across screen reloads. The components allow one to implement functionality like a sidebar or menu items. Third-party plugins can extend them by default.

Installation

Install the module

```
npm install @wordpress/interface --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.*

API Usage

Complementary Areas

This component was named after a complementary landmark – a supporting section of the document, designed to be complementary to the main content at a similar level in the DOM hierarchy, but remains meaningful when separated from the main content.

`ComplementaryArea` and `ComplementaryArea.Slot` form a slot fill pair to render complementary areas. Multiple `ComplementaryArea` components representing different complementary areas may be rendered at the same time, but only one appears on the slot depending on which complementary area is enabled.

It is possible to control which complementary is enabled by using the store:

Below are some examples of how to control the active complementary area using the store:

```
wp.data
    .select( 'core/interface' )
    .getActiveComplementaryArea( 'core' );
// -> "edit-post/document"

wp.data
    .dispatch( 'core/interface' )
    .enableComplementaryArea( 'core', 'edit-post/block' );

wp.data
    .select( 'core/interface' )
    .getActiveComplementaryArea( 'core' );
// -> "edit-post/block"

wp.data
    .dispatch( 'core/interface' )
    .disableComplementaryArea( 'core' );

wp.data
    .select( 'core/interface' )
    .getActiveComplementaryArea( 'core' );
// -> null
```

Pinned Items

`PinnedItems` and `PinnedItems.Slot` form a slot fill pair to render pinned items (or areas) that act as a list of favorites similar to browser extensions listed in the Chrome Menu.

Example usage: `ComplementaryArea` component makes use of `PinnedItems` and automatically adds a pinned item for the complementary areas marked as a favorite.

```
wp.data.select( 'core/interface' ).isItemPinned( 'core', 'edit-post-block-patterns/block-patterns-sidebar' );
// -> false

wp.data.dispatch( 'core/interface' ).pinItem( 'core', 'edit-post-block-patterns/block-patterns-sidebar' );

wp.data.select( 'core/interface' ).isItemPinned( 'core', 'edit-post-block-patterns/block-patterns-sidebar' );
// -> true

wp.data.dispatch( 'core/interface' ).unpinItem( 'core', 'edit-post-block-patterns/block-patterns-sidebar' );

wp.data.select( 'core/interface' ).isItemPinned( 'core', 'edit-post-block-patterns/block-patterns-sidebar' ); -> false
```

Preferences

The interface package provides some helpers for implementing editor preferences.

Features

Features are boolean values used for toggling specific editor features on or off.

Set the default values for any features on editor initialization:

```
import { dispatch } from '@wordpress/data';
import { store as interfaceStore } from '@wordpress/interface';

function initialize() {
    // ...

    dispatch( interfaceStore ).setFeatureDefaults(
        'namespace/editor-or-plugin-name',
```

```
        {
            myFeatureName: true,
        }
    );

    // ...
}
```

Use the `toggleFeature` action and the `isFeatureActive` selector to toggle features within your app:

```
wp.data
    .select( 'core/interface' )
    .isFeatureActive( 'namespace/editor-or-plugin-name', 'myFeatureName' ); // true
wp.data
    .dispatch( 'core/interface' )
    .toggleFeature( 'namespace/editor-or-plugin-name', 'myFeatureName' );
wp.data
    .select( 'core/interface' )
    .isFeatureActive( 'namespace/editor-or-plugin-name', 'myFeatureName' ); // false
```

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 18:19:22 UTC
□ □□□ □□□ 2026-10-31 18:19:24 UTC