@wordpress/experiments

- Getting started
- Shipping experimental APIs
- Technical limitations
- Contributing to this package

`@wordpress/experiments` enables sharing private `__experimental` APIs across `@wordpress` packages without

publicly exposing them to WordPress extenders.

Getting started

Every `@wordpress` package wanting to privately access or expose experimental APIs must opt-in to `@wordpress/experiments`:

```
// In packages/block-editor/experiments.js:
import { __dangerousOptInToUnstableAPIsOnlyForCoreModules } from '@wordpress/experiments';
export const { lock, unlock } =
    __dangerousOptInToUnstableAPIsOnlyForCoreModules(
        'I know using unstable features means my plugin or theme will inevitably break on the
next WordPress release.',
        '@wordpress/block-editor' // Name of the package calling
__dangerousOptInToUnstableAPIsOnlyForCoreModules,
                                  // (not the name of the package whose APIs you want to
access)
    );
```

Each package may only opt in once. The function name communicates that plugins are not supposed to use it.

The function will throw an error if the following conditions are not met:

1. The first argument must exactly match the required consent string: `'I know using unstable features means my plugin or theme will inevitably break on the next WordPress release.'`.
2. The second argument must be a known `@wordpress` package that hasn't yet opted into `@wordpress/experiments`

Once the opt-in is complete, the obtained `lock()` and `unlock()` utilities enable hiding `__experimental` APIs from the naked eye:

```
// Say this object is exported from a package:
export const publicObject = {};

// However, this string is internal and should not be publicly available:
const __experimentalString = '__experimental information';

// Solution: lock the string "inside" of the object:
lock( publicObject, __experimentalString );

// The string is not nested in the object and cannot be extracted from it:
console.log( publicObject );
// {}

// The only way to access the string is by "unlocking" the object:
console.log( unlock( publicObject ) );
// "__experimental information"

// lock() accepts all data types, not just strings:
export const anotherObject = {};
lock( anotherObject, function __experimentalFn() {} );
console.log( unlock( anotherObject ) );
// function __experimentalFn() {}
```

Use `lock()` and `unlock()` to privately distribute the `__experimental` APIs across `@wordpress` packages:

```
// In packages/package1/index.js:
import { lock } from './experiments';

export const experiments = {};
/* Attach private data to the exported object */
lock(experiments, {
```

```
    __experimentalFunction: function() {},
});

// In packages/package2/index.js:
import { experiments } from '@wordpress/package1';
import { unlock } from './experiments';

const {
    __experimentalFunction
} = unlock( experiments );
```

Shipping experimental APIs

See the Experimental and Unstable APIs chapter of Coding Guidelines to learn how `lock()` and `unlock()` can help
you ship private experimental functions, arguments, components, properties, actions, selectors.

Technical limitations

A determined developer who would want to use the private experimental APIs at all costs would have to:

- Realize a private importing system exists
- Read the code where the risks would be spelled out in capital letters
- Explicitly type out he or she is aware of the consequences
- Pretend to register a `@wordpress` package (and trigger an error as soon as the real package is loaded)

Dangerously opting in to using these APIs by theme and plugin developers is not recommended. Furthermore, the WordPress Core philosophy to strive to maintain backward compatibility for third-party developers **does not apply** to experimental APIs registered via this package.

The consent string for opting in to these APIs may change at any time and without notice. This change will break existing third-party code. Such a change may occur in either a major or minor release.

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 18:20:55 UTC
□ □□□ □□□ 2026-10-31 18:20:56 UTC