ConfirmDialog

- Uncontrolled mode
- Controlled mode
- Props
  - __experimentalHideHeader
  - children
  - confirmButtonText
  - cancelButtonText
  - isOpen
  - isBusy
  - onConfirm
  - onCancel
  - size
  - title

See the WordPress Storybook for more detailed, interactive documentation.

`ConfirmDialog` is built on top of `Modal`
and displays a confirmation dialog, with *confirm* and *cancel* buttons.

The dialog is confirmed by clicking the *confirm* button or by pressing the `Enter` key.
It is cancelled (closed) by clicking the *cancel* button, by pressing the `ESC` key, or by clicking outside the dialog focus (i.e. the overlay).

`ConfirmDialog` has two main modes: controlled and uncontrolled.

Uncontrolled mode

Allows the component to be used standalone, just by declaring it as part of another React component's render method:

- It will be automatically open (displayed) upon mounting.

- It will be automatically closed when clicking the *cancel* button, by pressing the `ESC` key, or by clicking outside the dialog focus (i.e. the overlay).
- `onCancel` is not mandatory but can be passed. Even if passed, the dialog will still be able to close itself.

Activating this mode is as simple as omitting the `isOpen` prop. The only mandatory prop in this case is the `onConfirm` callback. The message is passed as the `children`.

```
import { __experimentalConfirmDialog as ConfirmDialog } from '@wordpress/components';

function Example() {
    return (
        <ConfirmDialog onConfirm={ () => console.debug( 'Confirmed!' ) }>
            Are you sure? <strong>This action cannot be undone!</strong>
        </ConfirmDialog>
    );
}
```

Controlled mode

Let the parent component control when the dialog is open/closed. It's activated when a boolean value is passed to `isOpen`:

- It will not be automatically closed. You need to let it know when to open/close by updating the value of the `isOpen` prop.
- Both `onConfirm` and the `onCancel` callbacks are mandatory props in this mode.
- You'll want to update the state that controls `isOpen` from the `onCancel` and `onConfirm` callbacks.

```
import { useState } from '@wordpress/element';
import { __experimentalConfirmDialog as ConfirmDialog } from '@wordpress/components';

function Example() {
    const [ isOpen, setIsOpen ] = useState( true );

    const handleConfirm = () => {
        console.debug( 'Confirmed!' );
        setIsOpen( false );
    };
```

```
    const handleCancel = () => {
        console.debug( 'Cancelled!' );
        setIsOpen( false );
    };

    return (
        <ConfirmDialog
            isOpen={ isOpen }
            onConfirm={ handleConfirm }
            onCancel={ handleCancel }
        >
            Are you sure? <strong>This action cannot be undone!</strong>
        </ConfirmDialog>
    );
}
```

Props

`__experimentalHideHeader`

- Type: `boolean`
- Required: No
- Default: `true`

Whether the underlying `Modal` should hide its header, `title` included.

`children`

- Type: `ReactNode`
- Required: Yes

The actual message for the dialog. It's passed as children and any valid `ReactNode` is accepted.

`confirmButtonText`

- Type: `string`
- Required: No

The optional custom text to display as the confirmation button’s label.

`cancelButtonText`

- Type: `string`
- Required: No

The optional custom text to display as the cancellation button’s label.

`isOpen`

- Type: `boolean`
- Required: No

Defines if the dialog is open (displayed) or closed (not rendered/displayed).
It also implicitly toggles the controlled mode if set or the uncontrolled mode if it’s not set.

`isBusy`

- Type: `boolean`
- Required: No

Indicates activity while an action is being performed.
When `true`, the confirm button will show a busy state.
Both buttons will be disabled.

`onConfirm`

- Type: `(event: DialogInputEvent) => void`
- Required: Yes

The callback that’s called when the user confirms.
A confirmation can happen when the `OK` button is clicked or when `Enter` is pressed.

`onCancel`

- Type: `((event: DialogInputEvent) => void)`
- Required: No

The callback that’s called when the user cancels. A cancellation can happen when the `Cancel` button is clicked, when the `ESC` key is pressed, or when a click outside of the dialog focus is detected (i.e. in the overlay).

It’s not required if `isOpen` is not set (uncontrolled mode), as the component will take care of closing itself, but you can still pass a callback if something must be done upon cancelling (the component will still close itself in this case).

If `isOpen` is set (controlled mode), then it’s required, and you need to set the state that defines `isOpen` to `false` as part of this callback if you want the dialog to close when the user cancels.

`size`

- Type: `"small" | "fill" | "medium" | "large"`
- Required: No

Size of the underlying Modal. See `Modal`‘s `size` prop.

`title`

- Type: `string`
- Required: No

The title of the underlying `Modal`: its accessible name, and the visible heading unless `__experimentalHideHeader` is `true`.

□□□ #2
□ □□□ □□□ 2026-10-31 18:16:38 UTC
□ □□□ □□□ 2026-10-31 18:16:39 UTC