ToolsPanel

- Development guidelines
  - ToolsPanel Layout
- Usage
- Props
  - hasInnerWrapper: boolean
  - dropdownMenuProps: {}
  - headingLevel: 1 | 2 | 3 | 4 | 5 | 6 | '1' | '2' | '3' | '4' | '5' | '6'
  - label: string
  - panelId: string | null
  - resetAll: (filters?: ResetAllFilter[]) => void
  - shouldRenderPlaceholderItems: boolean

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

These panels provide progressive discovery options for their children. For example the controls provided via block supports.

Development guidelines

The `ToolsPanel` creates a container with a header including a dropdown menu. The menu is generated automatically from the panel’s children matching the `ToolsPanelItem` component type.

Each menu item allows for the display of the corresponding child to be toggled on or off. The control’s `onSelect` and `onDeselect` callbacks are fired allowing for greater control over the child e.g. resetting block attributes when a block support control is toggled off.

Whether a child control is initially displayed or not is dependent upon if there has previously been a value set or the child has been flagged as displaying by default through the `isShownByDefault` prop. Determining whether a

child has a value is done via the `hasValue` function provided through the child's props.

Components that are not wrapped within a `ToolsPanelItem` are still rendered however they will not be represented within, or controlled by, the `ToolsPanel` menu. An example scenario that benefits from this could be displaying introduction or help text within a panel.

ToolsPanel Layout

The `ToolsPanel` has a two-column grid layout. By default, `ToolsPanelItem` components within the panel are styled to span both columns as this fits the majority of use-cases. Most non-control elements, such as help text, will be rendered as children of the related control's `ToolsPanelItem` and not require additional styling.

Suppose an element is related to multiple controls (e.g. a contrast checker), or the panel itself (e.g. a panel description). In that case, these will be rendered into the panel without a wrapping `ToolsPanelItem`. They'll then only span a single column by default. If this is undesirable, those elements will likely need a small style tweak, e.g. `grid-column: 1 / -1;`

The usage example below will illustrate a non-`ToolsPanelItem` description paragraph, controls that should display in a single row, and others spanning both columns.

Usage

```
import {
    BoxControl,
    __experimentalToolsPanel as ToolsPanel,
    __experimentalToolsPanelItem as ToolsPanelItem,
    __experimentalUnitControl as UnitControl,
} from '@wordpress/components';
import { useState } from '@wordpress/element';
import { __ } from '@wordpress/i18n';
import styles from './style.module.scss';

export function DimensionPanel() {
    const [ height, setHeight ] = useState();
```

```
const [ width, setWidth ] = useState();
const [ padding, setPadding ] = useState();
const [ margin, setMargin ] = useState();

const resetAll = () => {
    setHeight( undefined );
    setWidth( undefined );
    setPadding( undefined );
    setMargin( undefined );
};

return (
    <ToolsPanel label={ __( 'Dimensions' ) } resetAll={ resetAll }>
        <div className={ styles[ 'panel-description' ] }>
            Select dimensions or spacing related settings from the menu for
            additional controls.
        </div>
        <ToolsPanelItem
            className={ styles[ 'single-column-item' ] }
            hasValue={ () => !! height }
            label={ __( 'Height' ) }
            onDeselect={ () => setHeight( undefined ) }
            isShownByDefault
        >
            <UnitControl
                __next40pxDefaultSize
                label={ __( 'Height' ) }
                onChange={ setHeight }
                value={ height }
            />
        </ToolsPanelItem>
        <ToolsPanelItem
            className={ styles[ 'single-column-item' ] }
            hasValue={ () => !! width }
            label={ __( 'Width' ) }
            onDeselect={ () => setWidth( undefined ) }
            isShownByDefault
        >
            <UnitControl
                __next40pxDefaultSize
```

```
                    label={ __( 'Width' ) }
                    onChange={ setWidth }
                    value={ width }
                />
            </ToolsPanelItem>
            <ToolsPanelItem
                hasValue={ () => !! padding }
                label={ __( 'Padding' ) }
                onDeselect={ () => setPadding( undefined ) }
            >
                <BoxControl
                    label={ __( 'Padding' ) }
                    onChange={ setPadding }
                    values={ padding }
                    allowReset={ false }
                />
            </ToolsPanelItem>
            <ToolsPanelItem
                hasValue={ () => !! margin }
                label={ __( 'Margin' ) }
                onDeselect={ () => setMargin( undefined ) }
            >
                <BoxControl
                    label={ __( 'Margin' ) }
                    onChange={ setMargin }
                    values={ margin }
                    allowReset={ false }
                />
            </ToolsPanelItem>
        </ToolsPanel>
    );
}
```

Add these styles to `style.module.scss`. The repeated selector keeps the single-column override above ToolsPanelItem's default full-width rule, regardless of stylesheet order.

```
.panel-description {
    grid-column: span 2;
}
```

```
.single-column-item.single-column-item {
    grid-column: span 1;
}
```

Props

`hasInnerWrapper`: `boolean`

Flags that the items in this ToolsPanel will be contained within an inner wrapper element allowing the panel to lay them out accordingly.

- Required: No
- Default: `false`

`dropdownMenuProps`: `{}`

The popover props to configure panel's `DropdownMenu`.

- Type: `DropdownMenuProps`
- Required: No

`headingLevel`: `1 | 2 | 3 | 4 | 5 | 6 | '1' | '2' | '3' | '4' | '5' | '6'`

The heading level of the panel's header.

- Required: No
- Default: `2`

`label`: `string`

Text to be displayed within the panel's header and as the `aria-label` for the panel's dropdown menu.

- Required: Yes

`panelId`: `string | null`

If a `panelId` is set, it is passed through the `ToolsPanelContext` and used to restrict panel items. When a `panelId` is set, items can only register themselves if the `panelId` is explicitly `null` or the item's `panelId` matches exactly.

- Required: No

`resetAll`: `( filters?: ResetAllFilter[] ) => void`

A function to call when the `Reset all` menu option is selected. As an argument, it receives an array containing the `resetAllFilter` callbacks of all the valid registered `ToolsPanelItems`.

- Required: Yes

`shouldRenderPlaceholderItems`: `boolean`

Advises the `ToolsPanel` that all of its `ToolsPanelItem` children should render placeholder content (instead of `null`) when they are toggled off and hidden.

Note that placeholder items won't apply the `className` that would be normally applied to a visible `ToolsPanelItem` via the `className` prop.

- Required: No
- Default: `false`

□□□ #2
□ □□□ □□□ 2026-10-31 18:16:27 UTC
□ □□□ □□□ 2026-10-31 18:16:29 UTC