UnitControl

- Usage
- Props
  - disableUnits: boolean
  - isPressEnterToChange: boolean
  - isResetValueOnUnitChange: boolean
  - isUnitSelectTabbable: boolean
  - label: string
  - labelPosition: string
  - onBlur: FocusEventHandler< HTMLInputElement | HTMLSelectElement >
  - onFocus: FocusEventHandler< HTMLInputElement | HTMLSelectElement >
  - onChange: UnitControlOnChangeCallback
  - onUnitChange: UnitControlOnChangeCallback
  - size: string
  - unit: string
  - units: WPUnitControlUnit[]
  - value: number | string

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

`UnitControl` allows the user to set a numeric quantity as well as a unit (e.g. `px`).

Usage

```
import { useState } from 'react';
import { __experimentalUnitControl as UnitControl } from '@wordpress/components';

const Example = () => {
```

```
    const [ value, setValue ] = useState( '10px' );

    return <UnitControl onChange={ setValue } value={ value } />;
};
```

Props

`disableUnits`: `boolean`

If true, the unit `<select>` is hidden.

- Required: No
- Default: `false`

`isPressEnterToChange`: `boolean`

If `true`, the `ENTER` key press is required in order to trigger an `onChange`. If enabled, a change is also triggered when tabbing away (`onBlur`).

- Required: No
- Default: `false`

`isResetValueOnUnitChange`: `boolean`

If `true`, and the selected unit provides a `default` value, this value is set when changing units.

- Required: No
- Default: `false`

`isUnitSelectTabbable`: `boolean`

Determines if the unit `<select>` is tabbable.

- Required: No
- Default: `true`

`label`: `string`

If this property is added, a label will be generated using label property as the content.

- Required: No

`labelPosition`: `string`

The position of the label (`top`, `side`, `bottom`, or `edge`).

- Required: No

`onBlur`: `FocusEventHandler< HTMLInputElement | HTMLSelectElement >`

Callback invoked when either the quantity or unit inputs fire the `blur` event.

- Required: No

`onFocus`: `FocusEventHandler< HTMLInputElement | HTMLSelectElement >`

Callback invoked when either the quantity or unit inputs fire the `focus` event.

- Required: No

`onChange`: `UnitControlOnChangeCallback`

Callback when the `value` changes.

- Required: No

`onUnitChange`: `UnitControlOnChangeCallback`

Callback when the `unit` changes.

- Required: No

`size`: `string`

Adjusts the size of the input.
Sizes include: `default`, `small`

- Required: No
- Default: `default`

`unit`: `string`

Deprecated: Current unit value.
Instead, provide a unit with a value through the `value` prop.

Example:

```
<UnitControl value="50%" />
```

- Required: No

`units`: `WPUnitControlUnit[]`

Collection of available units.

- Required: No

Example:

```
import { useState } from 'react';
import { __experimentalUnitControl as UnitControl } from '@wordpress/components';

const Example = () => {
    const [ value, setValue ] = useState( '10px' );

    const units = [
        { value: 'px', label: 'px', default: 0 },
        { value: '%', label: '%', default: 10 },
        { value: 'em', label: 'em', default: 0 },
    ];

    return (
        <UnitControl onChange={ setValue } value={ value } units={ units } />
    );
};
```

A `default` value (in the example above, `10` for `%`), if defined, is set as the new `value` when a unit changes. This is helpful in scenarios where changing a unit may cause drastic results, such as changing from `px` to `vh`.

`value`: `number | string`

Current value. If passed as a string, the current unit will be inferred from this value.
For example, a `value` of `50%` will set the current unit to `%`.

Example:

```
<UnitControl value="50%" />
```

- Required: No

□□□ #2
□ □□□ □□□ 2026-10-31 18:15:23 UTC
□ □□□ □□□ 2026-10-31 18:15:25 UTC