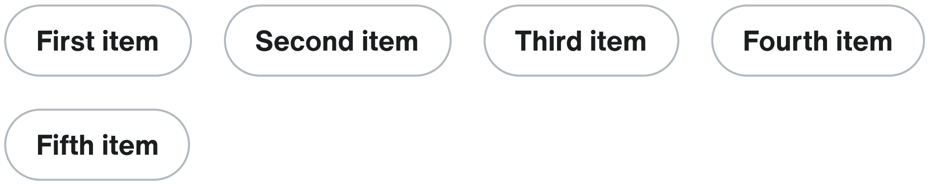
TELUS Component Library

ButtonGroup

Multi-Platform Component | Figma UI KIT



```
<ButtonGroup
  items={[
    { label: 'First item', id: 'first' },
    { label: 'Second item', id: 'second' },
    { label: 'Third item', id: 'third' },
    { label: 'Fourth item', id: 'fourth' },
    { label: 'Fifth item', id: 'fifth' }
  ]}
/>
```

Introduction

ButtonGroup allows the selection of (by default) one element from a selection of labelled buttons.

Spacing between buttons increases on wider viewports (`lg` and `xl`).

Follow the appropriate instructions to add this component in to your app.

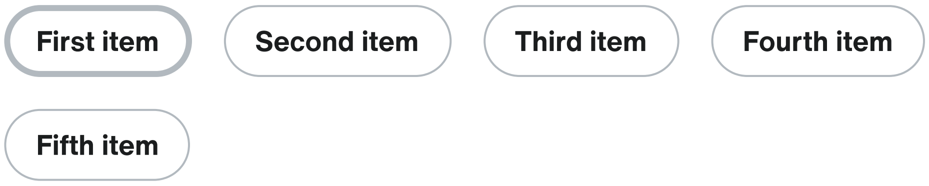
Guidance

Use for sets of options with very simple, short labels, ideally one word or two.

Consider `RadioGroup` for items with labels of a few words in length and `RadioCard` for more complex options requiring a title and additional description content.

`maxValues` prop

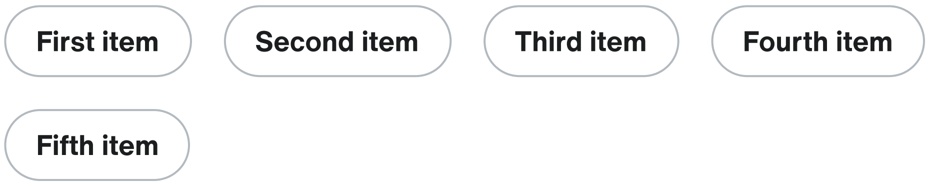
The prop `maxValues` may be passed to allow more than one item to be selected. Pass a number to set that as the maximum, or `null` to have no maximum and allow any number of selections.



```
<ButtonGroup
  items={[
    { label: 'First item', id: 'first' },
    { label: 'Second item', id: 'second' },
    { label: 'Third item', id: 'third' },
    { label: 'Fourth item', id: 'fourth' },
    { label: 'Fifth item', id: 'fifth' }
  ]}
  maxValues={2}
/>
```

`onChange` prop

An `onChange` function may be provided, which is called when a ButtonGroup item is toggled.



```
<ButtonGroup
  items={[
    { label: 'First item', id: 'first' },
    { label: 'Second item', id: 'second' },
    { label: 'Third item', id: 'third' },
    { label: 'Fourth item', id: 'fourth' },
    { label: 'Fifth item', id: 'fifth' }
  ]}
  onChange={(state) => setTimeout(() => alert(`ButtonGroup has selected: ${state}`), 100)}
/>
```

Controlled usage with `values` prop

Pass an array of item ids to the `values` prop as well as `onChange` to control the ButtonGroup from a parent.

This is usually the easiest way to control sibling content with a ButtonGroup.

No selection saved

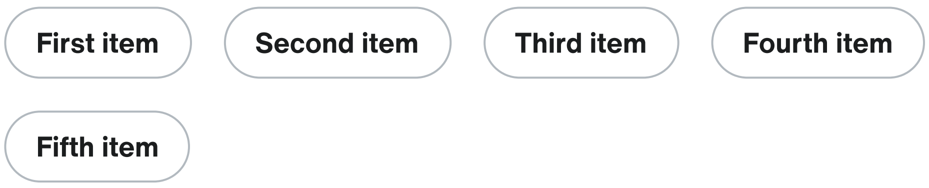
First item **Second item** **Third item** **Fourth item**

Fifth item

```
function ControlledButtonGroup() {
  const [values, setValues] = useState([])
  const text = values.length ? `Saved: "${values.join('", "')}".` : 'No selection saved'
  return (
    <StackView space={2}>
      <Typography variant={{ size: 'small' }}>{text}</Typography>
      <ButtonGroup
        values={values}
        onChange={setValues}
        items={[
          { label: 'First item', id: 'first' },
          { label: 'Second item', id: 'second' },
          { label: 'Third item', id: 'third' },
          { label: 'Fourth item', id: 'fourth' },
          { label: 'Fifth item', id: 'fifth' }
        ]}
      />
    </StackView>
  )
}
```

Ensuring at least one item should be selected

You can create your own validation logic to ensure at least one item is selected, see a suggestion below.



```
function ControlledButtonGroup() {
  const [values, setValues] = useState([])
  const [text, setText] = useState('')
  const handleChange = (values) => {
    if (values.length > 0) {
      setValues(values)
      setText(`Saved: "${values.join('", "')}".`)
    } else {
      setText('Please select at least one value')
    }
  }

  return (
    <StackView space={2}>
      <Typography variant={{ size: 'small' }}>{text}
</Typography>
      <ButtonGroup
        values={values}
        onChange={handleChange}
        items={[
          { label: 'First item', id: 'first' },
          { label: 'Second item', id: 'second' },
          { label: 'Third item', id: 'third' },
          { label: 'Fourth item', id: 'fourth' },
          { label: 'Fifth item', id: 'fifth' }
        ]}
      />
    </StackView>
  )
}
```

Ensuring at least one item should be selected

You can define items to be selected by default by passing them as argument of the `useState` hook during the creation of the `values` array. The example below sets the second item as selected by default.

Saved: "second".

First item Second item Third item Fourth item

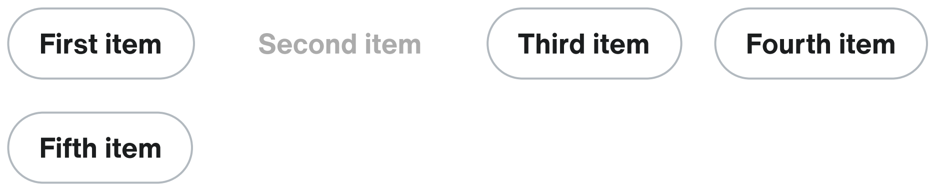
Fifth item

```
function ControlledButtonGroup() {
  const [values, setValues] = useState(['second'])
  const text = values.length ? `Saved: "${values.join('",
"')}".` : 'No selection saved'
  return (
    <StackView space={2}>
      <Typography variant={{ size: 'small' }}>{text}
</Typography>
      <ButtonGroup
        values={values}
        onChange={setValues}
        items={[
          { label: 'First item', id: 'first' },
          { label: 'Second item', id: 'second' },
          { label: 'Third item', id: 'third' },
          { label: 'Fourth item', id: 'fourth' },
          { label: 'Fifth item', id: 'fifth' }
        ]}
      />
    </StackView>
  )
}
```

Uncontrolled usage

The ButtonGroup manages its own state internally if `values` is not passed.

A default selection may be set by passing an array of item IDs to `initialValues`. Do not use both `initialValues` and `values` props in the same `ButtonGroup`.



```
<ButtonGroup
  items={[
    { label: 'First item', id: 'first' },
    { label: 'Second item', id: 'second' },
    { label: 'Third item', id: 'third' },
    { label: 'Fourth item', id: 'fourth' },
    { label: 'Fifth item', id: 'fifth' }
  ]}
  initialValues={['second']}
/>
```

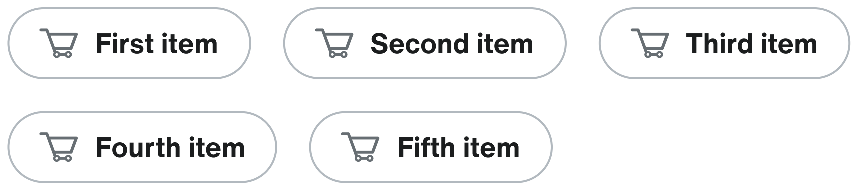
Usage with icon props

To add icons on each `item` of a `ButtonGroup`:

- Import the icons you want to use.

```
import { ExampleIcon } from '@telus-uds/palette-
allium/build/web/icons'
```

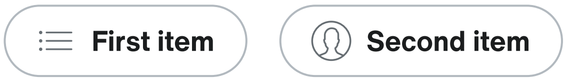
- Add the property `icon` with the imported icon name as the value `icon: ExampleIcon` on each item and it should appear on left side of the label.
- You can use the `IconPosition` prop on the component to set the position of all icons to the right side of the labels.



```
<ButtonGroup
  items={[
    { label: 'First item', id: 'first', icon: ExampleIcon },
    { label: 'Second item', id: 'second', icon: ExampleIcon },
    { label: 'Third item', id: 'third', icon: ExampleIcon },
    { label: 'Fourth item', id: 'fourth', icon: ExampleIcon },
    { label: 'Fifth item', id: 'fifth', icon: ExampleIcon }
  ]}
  iconPosition="left"
/>
```

ⓘ INFO

We use a icon wrapper called `Icons` in the documentation to avoid import conflicts. You can replace `ExampleIcon` by the combination `Icons.IconName` to view any palette icon on the code playground.



```
<ButtonGroup
  items={[
    { label: 'First item', id: 'first', icon: Icons.List },
    { label: 'Second item', id: 'second', icon: Icons.User }
  ]}
/>
```

Accessibility

By default, `ButtonGroup` is treated by accessibility tools as a group of radio buttons.

If `maxValues` prop is used to allow multiple selections, the buttons will be treated like checkboxes.

Platform considerations

The component is available on both native platforms and web.

Props

Name	Type	Platform	Default	Description
tokens	tokens	standard		System tokens prop, see tokens for more details
variant	variant	standard		System variant prop, see variants for more details
maxValues	number	standard		The maximum number of items a user may select at once. Defaults to 1 and behaves like radio buttons. To have no limit and allow any number of selections, pass \`null\`.
items	arrayOf	standard		The options a user may select
onChange	func	standard		If provided, this function is called when the current selection is changed and is passed an array of the \`id\`s of all currently selected \`items\`.
values	arrayOf	standard		If the selected item(s) in the button group are to

Name	Type	Platform	Default	Description
				be controlled externally by a parent component, pass an array of strings as well as an `onChange` handler. Passing an array for "values" makes the ButtonGroup a "controlled" component that expects its state to be handled via `onChange` and so doesn't handle it itself.
initialValues	arrayOf	standard		If `values` is not passed, making the ButtonGroup an "uncontrolled" component managing its own selected state, a default set of selections may be provided. Changing the `initialValues` does not change the user's selections.
iconPosition	'left' | 'right'	standard		Defines if the icon will be displayed on the right or left side of the label.

Name	Type	Platform	Default	Description
legend	string	standard		Main text used to describe this group, used in Fieldset's Legend element.
hint	string	standard		Optional additional text giving more detail to help a user make a choice.
hintPosition	'inline' | 'below'	standard		Position of the hint relative to label. Use `below` to display a larger hint below the label.
tooltip	string	standard		Optional tooltip text content to include alongside the legend and hint.
validation	'error' | 'success'	standard		Current validation status of the group, passed to the feedback element if there is one.
feedback	string	standard		If provided, a Feedback element is rendered containing this text.
readOnly	bool	standard		If true, the buttons cannot be selected by the user and

Name	Type	Platform	Default	Description
				simply show their current state.
inactive	bool	standard		If true, the buttons cannot be interacted with, elements are set as `disabled` and if the theme supports `inactive` appearances rules, these are applied.
name	string	standard		On Web, this is passed to the `name` attribute of the fieldset.
copy	'en' | 'fr'	standard		Sets the language of microcopy in subcomponents (e.g. Tooltip's default accessibility label).

Tokens

In exceptional circumstances, the following tokens can be passed to this component to override its default styles. **Do not do this unless absolutely necessary.** Read more about overriding styles.

▶ View Tokens

Variants

This component does not have any stylistic variants.

Feedback

- Spotted a problem with this component? Raise an issue on GitHub
- See any existing issues for this component
- Contact the team on slack in #ds-support