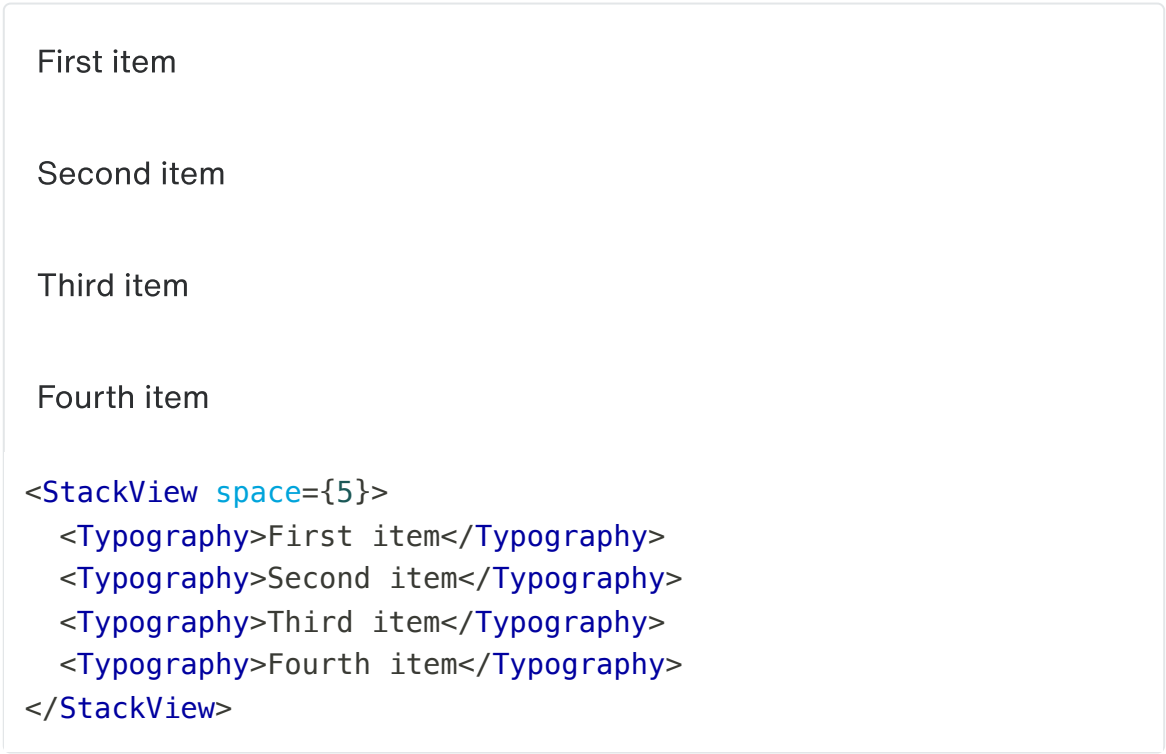
TELUS Component Library

StackView

Multi-Platform Component




```
<StackView space={5}>
  <Typography>First item</Typography>
  <Typography>Second item</Typography>
  <Typography>Third item</Typography>
  <Typography>Fourth item</Typography>
</StackView>
```

Introduction

StackView puts space evenly between its children.

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

Guidance

Use `StackView` to create layouts with simple, neat, even spacing.

- If a layout requires different spacing between different items, use `Spacer`.
- If the items need to wrap if they exceed a fixed size, use `StackWrap`.
- For space *around* an element or group of elements, wrap them in a `Box`.
- For more complex grids designed using a Bootstrap-like 12-column-grid system, consider `FlexGrid`.

`space` prop

The `space` prop sets the size of spacing between items, using the **spacing scale**.

Simple numbers may be used, or objects (see spacing scale docs for details. These examples change spacing depending on the viewport.

First item

Second item

Third item

Fourth item

```
<StackView space={{ xs: 1, md: 3, xl: 5 }}>
  <Typography>First item</Typography>
  <Typography>Second item</Typography>
  <Typography>Third item</Typography>
  <Typography>Fourth item</Typography>
</StackView>
```

`direction` prop

By default, items are stacked vertically as a flex column. Pass `direction` as `'row'` to stack horizontally. This applies `flexDirection: 'row'` styles to the container and passes appropriate props to spacers.

First item Second item Third item Fourth item

```
<StackView direction="row" space={{ xs: 1, md: 3, xl: 5 }}>
  <Typography>First item</Typography>
  <Typography>Second item</Typography>
  <Typography>Third item</Typography>
  <Typography>Fourth item</Typography>
</StackView>
```

Directions of `row-reverse` and `column-reverse` may be used to reverse the visual display of the stacked children.

The `direction` prop may also be passed a responsive object keyed by viewports.

First item Second item Third item Fourth item

```
<StackView space={4} direction={{ xs: 'column', md: 'row' }}>
  <Typography>First item</Typography>
  <Typography>Second item</Typography>
  <Typography>Third item</Typography>
  <Typography>Fourth item</Typography>
</StackView>
```

`divider` prop

By default, space is created by inserting a `<Spacer>` component between children.

If the `divider` prop is passed as true, children are instead spaced with a `Divider`. The StackView's `space` prop value is passed down to these dividers, rendering spacers before and after the dividing line.

First item

Second item

Third item

Fourth item

```
<StackView space={{ xs: 1, md: 3, xl: 5 }} divider>
  <Typography>First item</Typography>
```

```
  <Typography>Second item</Typography>
  <Typography>Third item</Typography>
  <Typography>Fourth item</Typography>
</StackView>
```

`direction` is applied to dividers automatically.

First item | Second item | Third item | Fourth item

```
<StackView space={{ xs: 1, md: 3, xl: 5 }} divider
direction="row">
  <Typography>First item</Typography>
  <Typography>Second item</Typography>
  <Typography>Third item</Typography>
  <Typography>Fourth item</Typography>
</StackView>
```

The `divider` prop may take an object of Divider props, which are passed to the rendered dividers.

First item

Second item

Third item

Fourth item

```
<StackView
  space={{ xs: 1, md: 3, xl: 5 }}
  divider={{ variant: { decorative: true, weight: 'thick' } }}
>
  <Typography>First item</Typography>
  <Typography>Second item</Typography>
  <Typography>Third item</Typography>
```

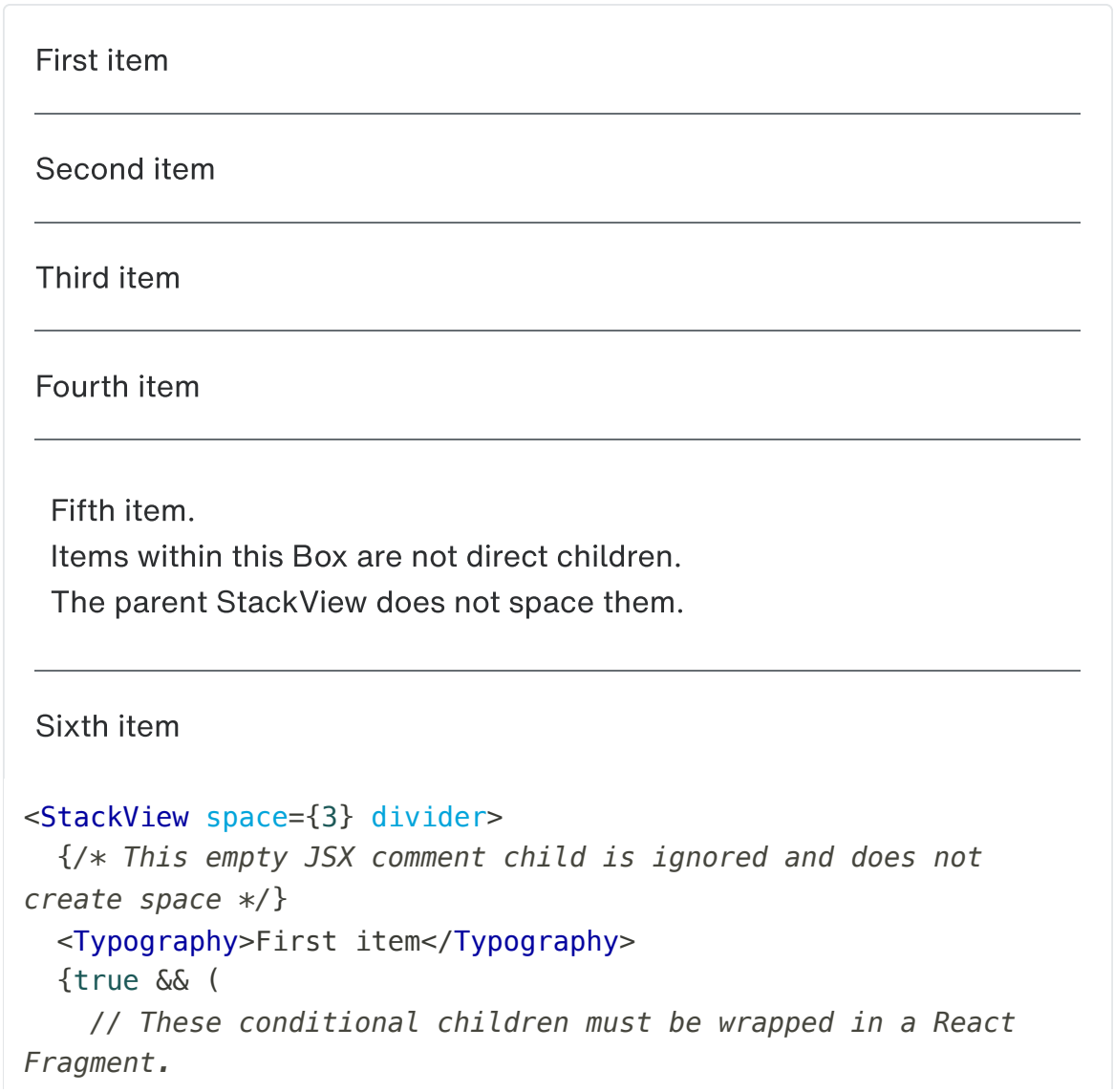
```
  <Typography>Fourth item</Typography>
</StackView>
```

Composition notes

`StackView` only spaces rendered, direct children. This means common React composition patterns may be used without needing to worry about creating spurious spacing:

- Comments and nullish children can be safely used and are ignored when spacing is inserted
- Fragments may be used when composing children, items within a fragment will still be spaced

This example demonstrates how StackView responds to various common React composition patterns:




```
<StackView space={3} divider>
  {/* This empty JSX comment child is ignored and does not
create space */}
  <Typography>First item</Typography>
  {true && (
    // These conditional children must be wrapped in a React
Fragment.
```

```
    // It does not prevent them from being spaced.
    <>
      <Typography>Second item</Typography>
      <Typography>Third item</Typography>
    </>
  )}
  {false && <Typography>Not rendered, no extra
space</Typography>}
  <Typography>Fourth item</Typography>
  <Box space={2} variant={{ background: 'light' }}>
    <Typography>Fifth item.</Typography>
    <Typography>Items within this Box are not direct children.
</Typography>
    <Typography>The parent StackView does not space them.
</Typography>
  </Box>
  <Typography>Sixth item</Typography>
</StackView>
```

Accessibility

By default, StackView applies no accessibility props or values relevant to accessibility tools. It may accept React Native accessibility props (web docs, native docs), and applies them to the outer container.

Semantic `divider`

If the `divider` prop is used, on Web, accessibility tools will be aware that a semantic separator exists between elements.

`Tag` prop for semantic HTML

By default, StackView renders a `<div>` on web and a `View` with no accessibility role on native apps.

To render a HTML attribute with a semantic meaning on web, pass a string naming a HTML semantic layout tag to the `tag` prop. An `accessibilityRole` may also be passed if a role is needed in native apps. Heading tags (`h1` , `h2` , `h3` , `h4` , `h5` , `h6`) are by default mapped to "heading" role in native apps.

See the `tag` entry in the props table for the full list of supported tags.

"h2" tag on web

This subheading is part of the semantic h2 heading

1: On web, "label"; in native apps, "header"

This is inside a "Section" tag on web.

On web, all the label's content serves as this section's accessible name.

This helps users of accessibility tools understand a page's structure.

In native apps, headers may be the closest equivalent to named sections.

This is inside a "footer" tag on web.

That footer is inside the "main" stack.

```
<StackView tag="main">
  <StackView tag="h2" space={2}>
    <Typography variant={{ size: 'h1' }}>"h2" tag on
web</Typography>
    <Typography variant={{ size: 'h4' }}>
      This subheading is part of the semantic h2 heading
    </Typography>
  </StackView>
  <StackView
    direction="row"
    tag="label"
    space={2}
    nativeID="label-stack"
    accessibilityRole="header"
  >
    <Typography variant={{ size: 'h3', color: 'secondary' }}>1:
</Typography>
    <Typography variant={{ size: 'h3' }}>On web, "label"; in
native apps, "header"</Typography>
  </StackView>
  <StackView tag="section" space={2}
accessibilityLabelledBy="label-stack">
    <Typography>This is inside a "Section" tag on web.
</Typography>
    <Typography>
      On web, all the label's content serves as this section's
accessible name.
    </Typography>
```

```
    <Typography>This helps users of accessibility tools
understand a page's structure.</Typography>
    <Typography>
      In native apps, headers may be the closest equivalent to
named sections.
    </Typography>
  </StackView>
  <StackView tag="footer" space={2}>
    <Typography>This is inside a "footer" tag on web.
</Typography>
    <Typography>That footer is inside the "main" stack.
</Typography>
  </StackView>
</StackView>
```

Platform considerations

The component is available on both native platforms and web.

Props

Name	Type	Platform	Default	Description
space	union	standard	0	The size of the spacer according to the theme's spacing scale. Either a number corresponding to a position on the theme's spacing scale (1 is smallest, 2 is second smallest, etc), or, a SpacingObject with viewport keys and options (see

Name	Type	Platform	Default	Description
				`useSpacingScale` for details).
direction	custom	standard	'column'	Sets the `flexDirection` of the container and, if `divider` is used, gives the Divider the appropriate direction.
tokens	tokens	standard		System tokens prop, see tokens for more details
variant	variant	standard		System variant prop, see variants for more details
divider	union	standard		If true, renders a UDS `Divider` component between each item. If an object is passed, this object is passes as props to each Divider.
tag	'h1' | 'h2' | 'h3' | 'h4' | 'h5' | 'h6' | 'article' | 'aside' | 'blockquote' | 'footer' | 'figure' |	standard		Optional semantic HTML tag, to apply to the container on web. Does not change styling. In native apps, if a header tag is provided ("h1",

Name	Type	Platform	Default	Description
	'form' | 'header' | 'ul' | 'li' | 'main' | 'nav' | 'section' | 'label'			"h2" etc) and accessibilityRole prop is not defined, the accessibilityRole will default to "heading".
children	node	standard		A StackView may take any children, but will have no effect if it is only passed one child or is passed children wrapped in a component. If necessary, children may be wrapped in one React Fragment.

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.

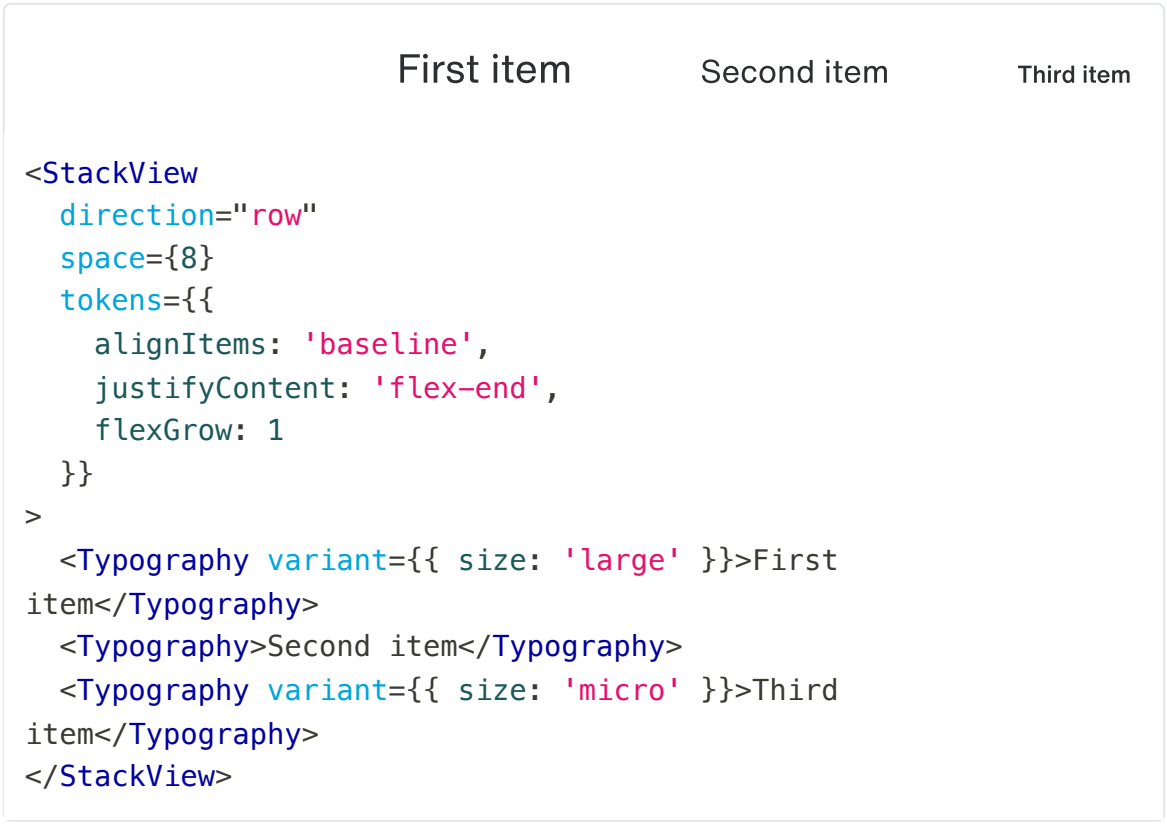
▶ View Tokens

Variants

This component does not have any stylistic variants.

To control flex alignment and positioning, the `tokens` prop may be passed an object to apply the flex layout styles `'alignItems'`, `'justifyContent'`, `'flexShrink'` and `'flexGrow'`.

As always with the `tokens` prop, this should be used only where strictly necessary to avoid creating off-brand layouts.




```
<StackView
  direction="row"
  space={8}
  tokens={{
    alignItems: 'baseline',
    justifyContent: 'flex-end',
    flexGrow: 1
  }}
>
  <Typography variant={{ size: 'large' }}>First
item</Typography>
  <Typography>Second item</Typography>
  <Typography variant={{ size: 'micro' }}>Third
item</Typography>
</StackView>
```

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