FlexGrid

Multi-Platform Component

1/4 2/4 3/4 4/4

```
<FlexGrid>
  <FlexGrid.Row>
    <FlexGrid.Col xs={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>1/4</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>2/4</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>3/4</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>4/4</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

Introduction

A mobile-first flexbox grid based on the "12-column grid" pattern.

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

Guidance

Use FlexGrid to build grid layouts based on divisions of 12: imagine the screen width as being divided into 12 slices: each FlexGrid row contains one or more columns which occupy a given number of those 12 slices.

A FlexGrid is built using a `<FlexGrid>` outer container which contains one or more `<FlexGrid.Row>`s, then each row contains one or more `<FlexGrid.Col>` columns, each of which claims a number of the 12 slices available for the overall row. The number of slices claimed may vary by viewport, allowing grids to be created that change shape based on the viewport.

None of the FlexGrid components depend on being direct children of each other and all may contain non-FlexGrid children, so FlexGrid may be used with other layout and spacing components such as Divider, Spacer, Box and StackView.

On the web, if you would like to reset the `zIndex` to `auto` for `FlexGrid`, `FlexGrid.Row`, and `FlexGrid.Column`, you can use `forceZIndex` flag available in the `themeOptions` prop on the `ThemeProvider` and pass its value as `true`

See below for detailed documentation on each of the subcomponents.

Three 4-slice columns	...that collapse to 12-slice full width	...when used on narrow screens.
Narrow column	...then a wider column that claims more space on wider screens.	

```
<FlexGrid outsideGutter={false}>
  <FlexGrid.Row>
    <FlexGrid.Col xs={12} md={4}>
      <Typography variant={{ size: 'small' }}>Three 4-slice
columns</Typography>
    </FlexGrid.Col>
    <FlexGrid.Col xs={12} md={4}>
      <Typography variant={{ size: 'small' }}>...that collapse
to 12-slice full width</Typography>
    </FlexGrid.Col>
    <FlexGrid.Col xs={12} md={4}>
      <Typography variant={{ size: 'small' }}>...when used on
narrow screens.</Typography>
```

```
    </FlexGrid.Col>
  </FlexGrid.Row>
  <FlexGrid.Row>
    <FlexGrid.Col xs={12}>
      <Divider space={2} />
    </FlexGrid.Col>
  </FlexGrid.Row>
  <FlexGrid.Row>
    <FlexGrid.Col xs={4} sm={3} lg={2}>
      <Typography variant={{ size: 'small' }}>Narrow
column</Typography>
    </FlexGrid.Col>
    <FlexGrid.Col xs={8} sm={9} lg={10}>
      <Typography variant={{ size: 'small' }}>
        ...then a wider column that claims more space on wider
screens.
      </Typography>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

Alternatives

For simpler or more flexible grid layouts, consider using StackView. Many grid layouts can be achieved by using `<StackView>` columns inside `<StackView direction="row">` rows, and StackView directions may be made conditional on viewport (for example, `<StackView direction={{ xs: 'column', md: 'row' }} />`).

For table grids where columns have the same width across different rows and where assisted-technology users may better understand the content based on column and row headers, consider Table (web only).

For grids where exactly 6 small captioned images follow a fixed responsive "waffle" pattern, consider WaffleGrid (web only).

Accessibility

Reversing the order of content using `*Reverse` props should be done sparingly as it creates a disconnect between the order of items seen visually on the screen and the order in which items are navigated by keyboard navigation and by screen readers. This may make pages harder to navigate or more

confusing for sighted users using keyboard navigation or partially-sighted users who use screen readers as an assistance rather than a replacement to scanning the screen visually.

`Tag` prop for semantic HTML

By default, FlexGrid 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.

Some text in a "footer".

This column also contains an "aside".

Some links

Inside of a "footer"

Containing a "nav"

```
<FlexGrid tag="footer">
  <FlexGrid.Row>
    <FlexGrid.Col xs={12} md={6}>
      <StackView tag="aside" space={2}>
        <Typography>Some text in a "footer".</Typography>
        <Typography>This column also contains an "aside".
</Typography>
      </StackView>
    </FlexGrid.Col>
    <FlexGrid.Col xs={12} md={6}>
      <StackView tag="nav" space={1}>
        <Link href="/">Some links</Link>
        <Link href="/">Inside of a "footer"</Link>
        <Link href="/">Containing a "nav"</Link>
      </StackView>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

Platform considerations

FlexGrid may be used in both web and native apps.

The prop `horizontalAlign` behaves somewhat differently between platforms:

- On web, text elements inside a column or row with the `horizontalAlign` prop applied will inherit left, right or center text alignment corresponding to that `horizontalAlign` prop, unless they either have explicitly set `textAlign` styles themselves or they have an ancestor below the FlexGrid that sets `textAlign`.
- In native apps, text elements do **not** inherit `textAlign` from the `horizontalAlign` FlexGrid element.

Subcomponents

A FlexGrid consists of:

- One FlexGrid container
- One or more FlexGrid.Row rows
- One or more FlexGrid.Col columns

`FlexGrid`

All instances of FlexGrid should be within one FlexGrid container. This sets the grid's outer container and provides all columns and rows with the `gutter` prop value.

`gutter`

By default, 16px of spacing is provided between all columns in all rows. The amount of spacing here is fixed and cannot be overridden; however, passing `false` to the `gutter` prop removes this default spacing from all columns in all rows.

1/3 2/3 3/3

```
<FlexGrid gutter={false}>
```

```
  <FlexGrid.Row>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>1/3</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>2/3</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>3/3</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

`outsideGutter`

The prop `outsideGutter` controls whether the 16px gutter should also be applied either side of the FlexGrid row's content. Passing `outsideGutter={false}` while keeping `gutter={true}` applies the spacing only between FlexGrid rows.

Do not set both `outsideGutter={false}` and `gutter={false}`; this will overcompensate and result in the FlexGrid's rows being wider than its parent.

`limitWidth`

By default, the FlexGrid sets the maximum width of the overall container to the maximum width of the current viewport, at viewports `sm` or above. To disable this behaviour and set no max width on the outer container at any viewport, set `limitWidth` to false.

Reverse props (`xsReverse`, `smReverse`...)

"Reverse" props may be set on the outer container for each viewport (`xsReverse`, `smReverse`, `mdReverse`, `lgReverse`, `xlReverse`). These control the order of the FlexGrid's rows.

Pass one of these props as `true` to reverse the rows at this viewport and above, and pass one as `false` to stop reversing at this viewport or above.

At different viewports

These rows do

Reorder themselves

```
<FlexGrid xsReverse={false} smReverse={true} xlReverse={false}>
  <FlexGrid.Row>
    <FlexGrid.Col xs={12}>
      <Typography>At different viewports</Typography>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Divider space={4} />
  <FlexGrid.Row>
    <FlexGrid.Col xs={12}>
      <Typography>These rows do</Typography>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Divider space={4} />
  <FlexGrid.Row>
    <FlexGrid.Col xs={12}>
      <Typography>Reorder themselves</Typography>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

`FlexGrid.Row`

Within the `FlexGrid` container, there should be one or more `FlexGrid.Row` rows.

horizontalAlign, verticalAlign and distribute props

These props set the flex alignment behaviours of the immediate children of the row - i.e. each `FlexGrid.Col` wrapper, and any elements placed in the row

not inside a `FlexGrid.Col`.

These props take a string and are not responsive - it is not expected that flex rows change their alignment styles based on the current viewport.

`horizontalAlign` and `distribute` apply horizontally and usually only have an effect when there is leftover space (i.e. when the columns total does not add up to `12`). `verticalAlign` applies vertically and usually only has an effect when columns in a row have different heights.

`horizontalAlign` values map to `justifyContent` and `textAlign` styles:

- `horizontalAlign="start"` maps to `{ justifyContent: 'flex-start', textAlign: 'left' }`
- `horizontalAlign="center"` maps to `{ justifyContent: 'center', textAlign: 'center' }`
- `horizontalAlign="end"` maps to `{ justifyContent: 'flex-end', textAlign: 'right' }`

Note that on native apps, due to React Native's limited text style inheritance, text within a FlexGrid.Row will not inherit these `textAlign` values and alignment should still be set on text components inside the row such as Typography.

horizontalAlign "start"

horizontalAlign "center"

horizontalAlign "end"

```
<FlexGrid>
  <FlexGrid.Row horizontalAlign="start">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>horizontalAlign</Typography>
      </Box>
    </FlexGrid.Col>
```

```
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>"start"</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Divider space={4} />
  <FlexGrid.Row horizontalAlign="center">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>horizontalAlign</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>"center"</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Divider space={4} />
  <FlexGrid.Row horizontalAlign="end">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>horizontalAlign</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>"end"</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

`distribute` values set the distribution of any leftover space and also map to `justifyContent`:

- `distribute="around"` distributes space evenly before, after and between each item. It maps to `{ justifyContent: 'space-around' }`
- `distribute="between"` distributes space evenly between items, leaving the first and last items tight against the left and right edges of the Row. It maps to `{ justifyContent: 'space-between' }`

If both `horizontalAlign` and `distribute` props are set, `distribute` has priority and the final `justifyContent` style of the rendered will be based on the `distribute` value not the `horizontalAlign` value.

distribute "between"

---

distribute "around"

```
<FlexGrid>
  <FlexGrid.Row distribute="between">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>distribute</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>"between"</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Divider space={4} />
  <FlexGrid.Row distribute="around">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>distribute</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>"around"</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

`verticalAlign` values map to `alignItems` styles:

- `verticalAlign="top"` maps to `{ alignItems: 'flex-start' }`
- `verticalAlign="middle"` maps to `{ alignItems: 'center' }`

- `verticalAlign="bottom"` maps to `{ alignItems: 'flex-end' }`

verticalAlign
"top"

Example

---

verticalAlign
"middle"

Example

---

verticalAlign
"bottom"

Example

```
<FlexGrid>
  <FlexGrid.Row verticalAlign="top">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} horizontal={2}
vertical={3}>
        <Typography>verticalAlign</Typography>
        <Typography>"top"</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} horizontal={2}
vertical={1}>
        <Typography variant={{ size: 'micro'
}}>Example</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Divider space={4} />
  <FlexGrid.Row verticalAlign="middle">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} horizontal={2}
vertical={3}>
        <Typography>verticalAlign</Typography>
        <Typography>"middle"</Typography>
      </Box>
```

```
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} horizontal={2}
vertical={1}>
        <Typography variant={{ size: 'micro'
}}>Example</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Divider space={4} />
  <FlexGrid.Row verticalAlign="bottom">
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} horizontal={2}
vertical={3}>
        <Typography>verticalAlign</Typography>
        <Typography>"bottom"</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4}>
      <Box variant={{ background: 'light' }} horizontal={2}
vertical={1}>
        <Typography variant={{ size: 'micro'
}}>Example</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

Reverse props (`xsReverse`, `smReverse`...)

"Reverse" props may be set on the outer container for each viewport (`xsReverse`, `smReverse`, `mdReverse`, `lgReverse`, `xlReverse`). These control the order of the columns within this row.

Pass one of these props as `true` to reverse the columns at this viewport and above, and pass one as `false` to stop reversing at this viewport or above.

`FlexGrid.Col`

Within each `FlexGrid.Row` row, there should be one or more `FlexGrid.Col` columns.

`xs`, `sm`, `md`, `lg` and `xl`

These props take number values, where the number indicates how many twelths of the available width this column tries to fill. For example, `3` means one quarter of the available width, `4` means a third, `6` means half, `9` means three quarters and `12` means the column tries to fill the whole width.

This applies at and above the viewport specified, until another prop is specified. For example, `<FlexGrid.Col xs={12} md={6} />` occupies the full width at viewports `xs` and `sm`, and occupies half the available width at viewports `md`, `lg` and `xl`.

If the values for all the columns in a row exceed `12`, the content of that row will wrap onto a new line. A common use of this prop is to create layouts that take up one row on wide screens and which collapse down to multiple rows on narrower screens.

First column Second column Third column

```
<FlexGrid gutter={false}>
  <FlexGrid.Row>
    <FlexGrid.Col xs={12} md={6} xl={4}>
      <Box right={{ xs: 0, md: 2 }} bottom={{ xs: 2, md: 0 }}>
        <Box variant={{ background: 'light' }} space={2}>
          <Typography variant={{ size: 'small' }}>First
column</Typography>
        </Box>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={12} md={6} xl={4}>
      <Box right={{ xs: 0, xl: 2 }} bottom={{ xs: 2, xl: 0 }}>
        <Box variant={{ background: 'light' }} space={2}>
          <Typography variant={{ size: 'small' }}>Second
column</Typography>
        </Box>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={12} md={12} xl={4}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography variant={{ size: 'small' }}>Third
column</Typography>
      </Box>
```

```
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

Offset props (`xsOffset` , `xsOffset` ...)

Offset props may be used to leave an amount of space empty before a column at and above the given viewport (`xsOffset` , `smOffset` , `mdOffset` , `lgOffset` , `xlOffset`).

Giving a column an offset prop gives similar visual output to preceding a column with an empty column that has the equivalent responsive prop. For example, `<FlexGrid.Col xsOffset={3} mdOffset={6}>` is similar to preceding a column with an empty column `<FlexGrid.Col xs={3} md={3} />` .

One Two Skip a few

Ninety-nine One hundred

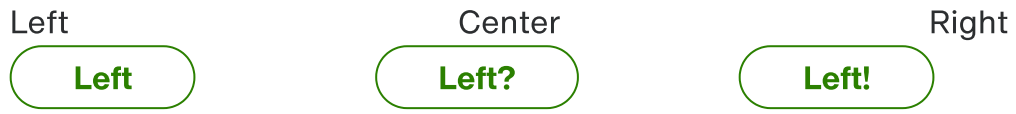
```
<FlexGrid>
  <FlexGrid.Row>
    <FlexGrid.Col xs={6} md={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography variant={{ size: 'small' }}>One</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={6} md={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography variant={{ size: 'small' }}>Two</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={6} xsOffset={6} md={3} mdOffset={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography variant={{ size: 'small' }}>Skip a
few</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
  <Spacer space={4} />
  <FlexGrid.Row>
    <FlexGrid.Col xs={6} md={3} xsOffset={6}>
```

```
      <Box variant={{ background: 'light' }} space={2}>
        <Typography variant={{ size: 'small' }}>Ninety-
nine</Typography>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={6} md={3}>
      <Box variant={{ background: 'light' }} space={2}>
        <Typography variant={{ size: 'small' }}>One
hundred</Typography>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

`horizontalAlign` prop

`FlexGrid.Col`'s `horizontalAlign` prop is quite different to `FlexGrid.Row`'s. It sets the text alignment of the wrapper (not `justifyContent`), and it is a responsive prop: it may take a string (`'left'`, `'center'` or `'right'`) or an object keyed by viewports (e.g. `{ xs: 'left', lg: 'right' }`).

Note that because this works using HTML text inheritance rules, it has no effect on child elements that already have a text alignment specified in their own styles, and it has no effect in native apps.




```
<FlexGrid>
  <FlexGrid.Row>
    <FlexGrid.Col xs={4} horizontalAlign="left">
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>Left</Typography>
        <Button variant={{ size: 'small' }} onPress={() => {}}>
          Left
        </Button>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4} horizontalAlign="center">
      <Box variant={{ background: 'light' }} space={2}>
```

```
        <Typography>Center</Typography>
        <Button variant={{ size: 'small' }} onPress={() => {}}>
          Left?
        </Button>
      </Box>
    </FlexGrid.Col>
    <FlexGrid.Col xs={4} horizontalAlign="right">
      <Box variant={{ background: 'light' }} space={2}>
        <Typography>Right</Typography>
        <Button variant={{ size: 'small' }} onPress={() => {}}>
          Left!
        </Button>
      </Box>
    </FlexGrid.Col>
  </FlexGrid.Row>
</FlexGrid>
```

Props

FlexGrid

Name	Type	Platform	Default	Description
limitWidth	bool	standard	true	Whether or not to give the grid a fixed width. This also centres the grid horizontally.
gutter	bool	standard	true	Whether or not to include gutters in between columns.
outsideGutter	bool	standard	true	Whether or not to include gutter

Name	Type	Platform	Default	Description
				at the ends to the left and right of the FlexGrid
xsReverse	bool	standard		Choose if the item order should be reversed from the 'xs' breakpoint. When you pass in false, the order will be normal from the xs breakpoint. By default, it inherits the behaviour set by the preceding prop.
smReverse	bool	standard		Choose if the item order should be reversed from the 'sm' breakpoint. When you pass in false, the order will be normal from the sm breakpoint. By default, it inherits the behaviour set by the preceding prop.

Name	Type	Platform	Default	Description
mdReverse	bool	standard		Choose if the item order should be reversed from the 'md' breakpoint. When you pass in false, the order will be normal from the md breakpoint. By default, it inherits the behaviour set by the preceding prop.
lgReverse	bool	standard		Choose if the item order should be reversed from the 'lg' breakpoint. When you pass in false, the order will be normal from the lg breakpoint. By default, it inherits the behaviour set by the preceding prop.
xlReverse	bool	standard		Choose if the item order should be

Name	Type	Platform	Default	Description
				reversed from the 'xl' breakpoint. When you pass in false, the order will be normal from the xl breakpoint. By default, it inherits the behaviour set by the preceding prop.
tag	'h1' | 'h2' | 'h3' | 'h4' | 'h5' | 'h6' | 'article' | 'aside' | 'blockquote' | 'footer' | 'figure' | 'form' | 'header' | 'ul' | 'li' | 'main' | 'nav' | 'section' | 'label'	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", "h2" etc) and accessibilityRole prop is not defined, the accessibilityRole will default to "heading".
children*	node	standard		The rows and columns of the Grid. Will typically be `FlexGrid.Row` and

Name	Type	Platform	Default	Description
				`FlexGrid.Col` components.

FlexGrid.Col

Name	Type	Platform	Default	Description
xs	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | true | false	standard		Specify number of columns within the 'xs' breakpoint range. `0` hides the column. `true` sets the column width automatically; `false` disables the prop
sm	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | true | false	standard		Specify number of columns within the 'sm' breakpoint range. `0` hides the column. `true` sets the column width automatically; `false` disables the prop
md	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |	standard		Specify number of columns within the 'md' breakpoint range. `0` hides the column. `true` sets the column

Name	Type	Platform	Default	Description
	true | false			width automatically; `false` disables the prop
lg	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | true | false	standard		Specify number of columns within the 'lg' breakpoint range. `0` hides the column. `true` sets the column width automatically; `false` disables the prop
xl	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | true | false	standard		Specify number of columns after the 'xl' breakpoint. `0` hides the column. `true` sets the column width automatically; `false` disables the prop
xsOffset	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11	standard		Offset the specified number of columns within the 'xs' breakpoint range. `0` hides the column.
smOffset	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7	standard		Offset the specified number of columns within

Name	Type	Platform	Default	Description
	| 8 | 9 | 10 | 11			the 'sm' breakpoint range.
mdOffset	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11	standard		Offset the specified number of columns within the 'md' breakpoint range.
lgOffset	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11	standard		Offset the specified number of columns within the 'lg' breakpoint range.
xlOffset	0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11	standard		Offset the specified number of columns within the 'xl' breakpoint range.
children*	node	standard		The columns of the Grid. Will typically be `FlexGrid.Col` components.
horizontalAlign	custom	standard		Align content horizontally within the column. Accepts a `PropType.string` following the [responsive prop](#/Layout?

Name	Type	Platform	Default	Description
				id=responsive) structure.
flex	bool	standard		(web only) Stretches the column to fill vertical space using `display: flex`. In native apps, FlexGrid.Col behaves as if this is always true (as do all `View`s).

FlexGrid.Row

Name	Type	Platform	Default	Description
horizontalAlign	'start' | 'center' | 'end'	standard		Align columns horizontally within their row.
verticalAlign	'top' | 'middle' | 'bottom'	standard		Align columns vertically within their row.
distribute	'around' | 'between'	standard		Distribute empty space around columns.
xsReverse	bool	standard		Choose if the item order should be reversed from the 'xs' breakpoint.

Name	Type	Platform	Default	Description
				When you pass in false, the order will be normal from the xs breakpoint. By default, it inherits the behaviour set by the preceding prop.
smReverse	bool	standard		Choose if the item order should be reversed from the 'sm' breakpoint. When you pass in false, the order will be normal from the sm breakpoint. By default, it inherits the behaviour set by the preceding prop.
mdReverse	bool	standard		Choose if the item order should be reversed from the 'md' breakpoint. When you pass in false, the order will be

Name	Type	Platform	Default	Description
				normal from the md breakpoint. By default, it inherits the behaviour set by the preceding prop.
lgReverse	bool	standard		Choose if the item order should be reversed from the 'lg' breakpoint. When you pass in false, the order will be normal from the lg breakpoint. By default, it inherits the behaviour set by the preceding prop.
xlReverse	bool	standard		Choose if the item order should be reversed from the 'xl' breakpoint. When you pass in false, the order will be normal from the xl breakpoint. By default, it

Name	Type	Platform	Default	Description
				inherits the behaviour set by the preceding prop.
children*	node	standard		

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