Grid

The responsive layout grid adapts to screen size and orientation, ensuring consistency across layouts.

The `Grid` component works well for a layout with a known number of columns. The columns can be configured with multiple breakpoints to specify the column span of each child.

View as Markdown | Feedback | Bundle size | Source | Material Design

How it works

The grid system is implemented with the `Grid` component:

- It uses CSS Flexbox (rather than CSS Grid) for high flexibility.
- The grid is always a flex item. Use the `container` prop to add a flex container.
- Item widths are set in percentages, so they're always fluid and sized relative to their parent element.
- There are five default grid breakpoints: xs, sm, md, lg, and xl. If you need custom breakpoints, check out custom breakpoints grid.
- You can give integer values for each breakpoint, to indicate how many of the 12 available columns are occupied by the component when the viewport width satisfies the breakpoint constraints.
- It uses the `gap` CSS property to add spacing between items.
- It does *not* support row spanning. Children elements cannot span multiple rows. We recommend using CSS Grid if you need this functionality.
- It does *not* automatically place children. It will try to fit the children one by one, and if there is not enough space, the rest of the children will start on the next line, and so on. If you need auto-placement, we recommend using CSS Grid instead.

⚠ The `Grid` component is a *layout* grid, not a *data* grid. If you need a data grid, check out the MUI X `DataGrid` component.

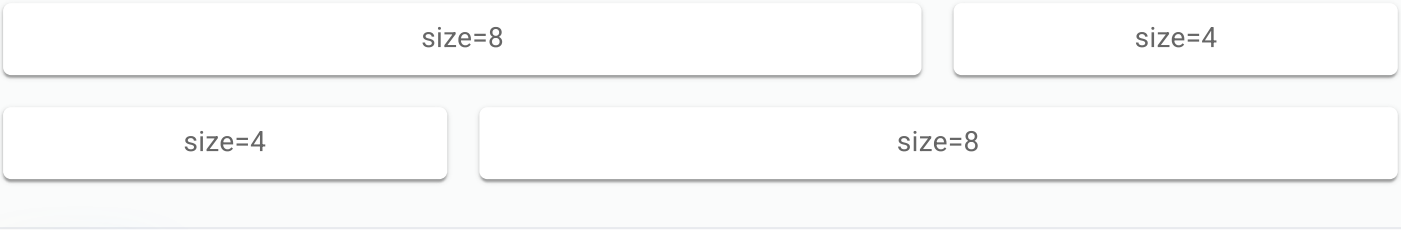
Fluid grids

Fluid grids use columns that scale and resize content. A fluid grid's layout can use breakpoints to determine if the layout needs to change dramatically.

Basic grid

In order to create a grid layout, you need a container. Use the `container` prop to create a grid container that wraps the grid items (the `Grid` is always an item).

Column widths are integer values between 1 and 12. For example, an item with `size={6}` occupies half of the grid container's width.



Edit in Chat JS TS Collapse code

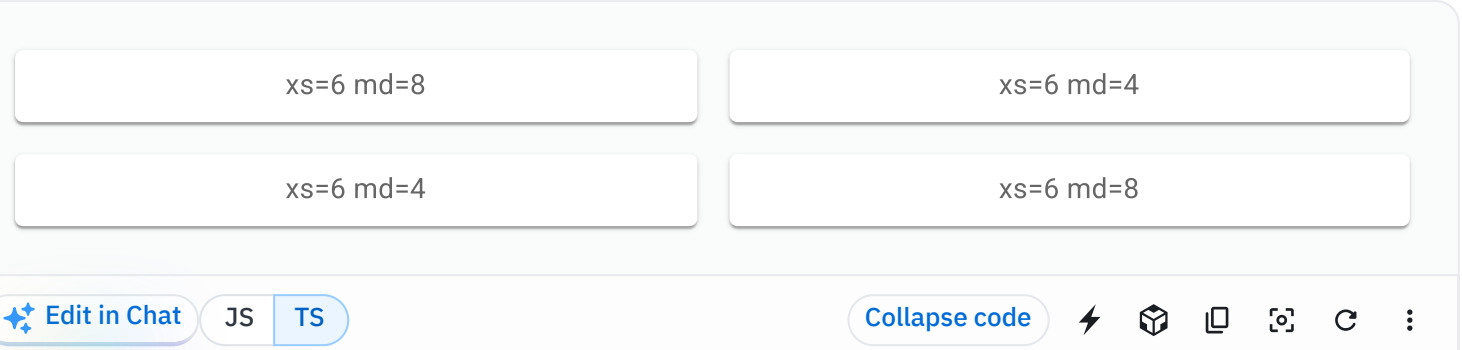
```tsx
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function BasicGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={2}>
        <Grid size={8}>
          <Item>size=8</Item>
        </Grid>
        <Grid size={4}>
          <Item>size=4</Item>
        </Grid>
        <Grid size={4}>
          <Item>size=4</Item>
        </Grid>
        <Grid size={8}>
          <Item>size=8</Item>
        </Grid>
      </Grid>
```

```
    </Box>
  );
}
```

Multiple breakpoints

Items may have multiple widths defined, causing the layout to change at the defined breakpoint. Width values apply to all wider breakpoints, and larger breakpoints override those given to smaller breakpoints.

For example, a component with `size={{ xs: 12, sm: 6 }}` occupies the entire viewport width when the viewport is less than 600 pixels wide. When the viewport grows beyond this size, the component occupies half of the total width—six columns rather than 12.



Edit in Chat | JS | TS | Collapse code

```
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function FullWidthGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={2}>
        <Grid size={{ xs: 6, md: 8 }}>
          <Item>xs=6 md=8</Item>
        </Grid>
        <Grid size={{ xs: 6, md: 4 }}>
          <Item>xs=6 md=4</Item>
        </Grid>
        <Grid size={{ xs: 6, md: 4 }}>
          <Item>xs=6 md=4</Item>
```

```
        </Grid>
        <Grid size={{ xs: 6, md: 8 }}>
          <Item>xs=6 md=8</Item>
        </Grid>
      </Grid>
    </Box>
  );
}
```

Spacing

Use the `spacing` prop to control the space between children. The spacing value can be any positive number (including decimals) or a string. The prop is converted into a CSS property using the `theme.spacing()` helper.

The following demo illustrates the use of the `spacing` prop:

spacing

○ 0 ○ 0.5 ○ 1 ◉ 2 ○ 3 ○ 4 ○ 8 ○ 12

```
<Grid container spacing={2}>
```

Copy

Row and column spacing

The `rowSpacing` and `columnSpacing` props let you specify row and column gaps independently of one another. They behave similarly to the `row-gap` and `column-gap` properties of CSS Grid.

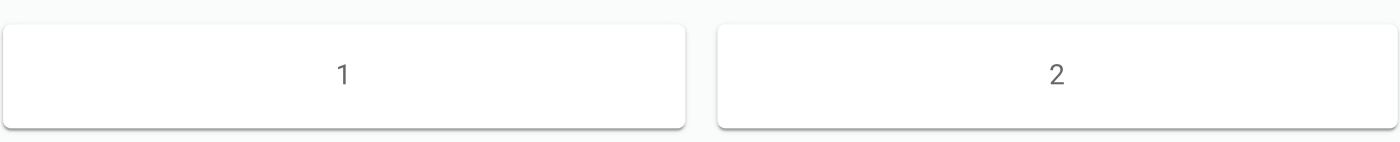
1	2
3	4

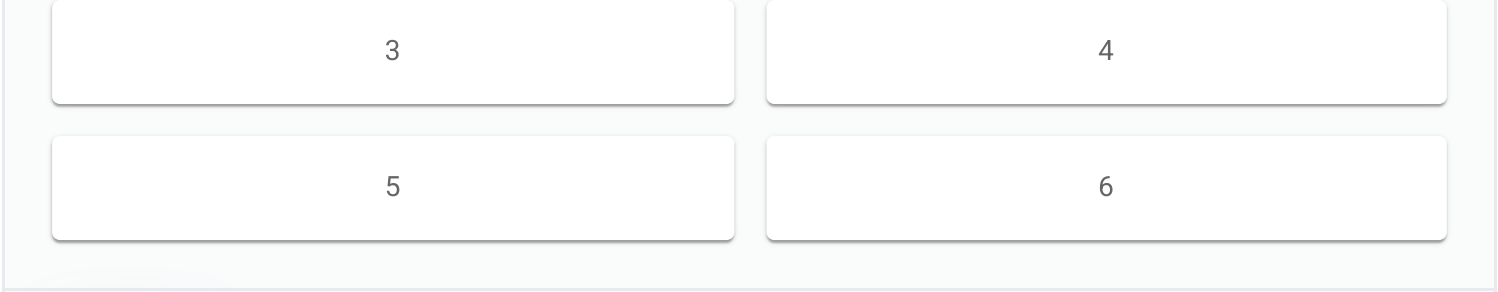
Edit in Chat JS TS Collapse code

```tsx
import { styled } from '@mui/material/styles';
import Grid from '@mui/material/Grid';
import Paper from '@mui/material/Paper';
import Box from '@mui/material/Box';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function RowAndColumnSpacing() {
  return (
    <Box sx={{ width: '100%' }}>
      <Grid container rowSpacing={1} columnSpacing={{ xs: 1, sm: 2, md: 3 }}>
        <Grid size={6}>
          <Item>1</Item>
        </Grid>
        <Grid size={6}>
          <Item>2</Item>
        </Grid>
        <Grid size={6}>
          <Item>3</Item>
        </Grid>
        <Grid size={6}>
          <Item>4</Item>
        </Grid>
      </Grid>
    </Box>
  );
}
```

Responsive values

You can set prop values to change when a given breakpoint is active. For instance, we can implement Material Design's recommended ↗ responsive layout grid, as seen in the following demo:





Edit in Chat JS TS Collapse code

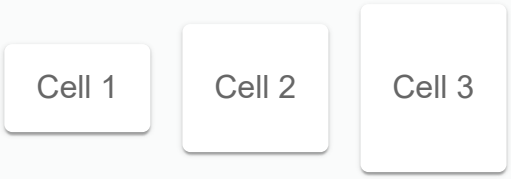
```tsx
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(2),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function ResponsiveGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={{ xs: 2, md: 3 }} columns={{ xs: 4, sm: 8, md: 12 }}>
        {Array.from(Array(6)).map((_, index) => (
          <Grid key={index} size={{ xs: 2, sm: 4, md: 4 }}>
            <Item>{index + 1}</Item>
          </Grid>
        ))}
      </Grid>
    </Box>
  );
}
```

Responsive values are supported by:

- `size`
- `columns`
- `columnSpacing`
- `direction`
- `rowSpacing`
- `spacing`
- `offset`

Interactive

Below is an interactive demo that lets you explore the visual results of the different settings:



direction

(•) row ( ) row-reverse

justifyContent

( ) flex-start (•) center ( ) flex-end ( ) space-between ( ) space-around

( ) space-evenly

alignItems

( ) flex-start (•) center ( ) flex-end ( ) stretch ( ) baseline

```
<Grid
  container
  direction="row"
  sx={{
    justifyContent: "center",
    alignItems: "center",
  }}
>
```

Copy

Auto-layout

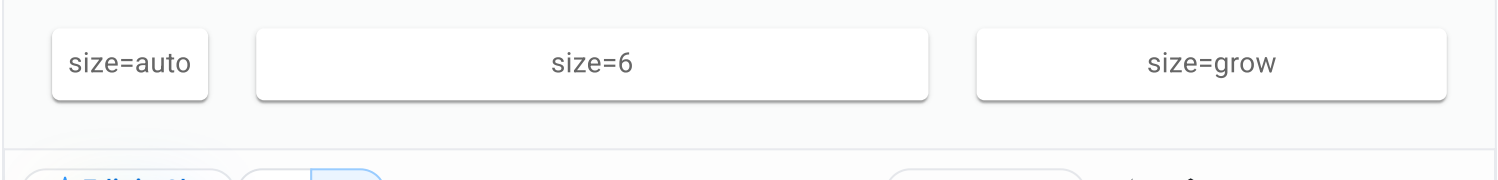
The auto-layout feature gives equal space to all items present. When you set the width of one item, the others will automatically resize to match it.

Edit in Chat JS TS Collapse code

```tsx
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function AutoGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={3}>
        <Grid size="grow">
          <Item>size=grow</Item>
        </Grid>
        <Grid size={6}>
          <Item>size=6</Item>
        </Grid>
        <Grid size="grow">
          <Item>size=grow</Item>
        </Grid>
      </Grid>
    </Box>
  );
}
```

Variable width content

When a breakpoint's value is given as `"auto"`, then a column's size will automatically adjust to match the width of its content. The demo below shows how this works:



Edit in Chat JS TS Collapse code

```jsx
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function VariableWidthGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={3}>
        <Grid size="auto">
          <Item>size=auto</Item>
        </Grid>
        <Grid size={6}>
          <Item>size=6</Item>
        </Grid>
        <Grid size="grow">
          <Item>size=grow</Item>
        </Grid>
      </Grid>
    </Box>
  );
}
```

Nested grid

The grid container that renders as a **direct child** inside another grid container is a nested grid that inherits its `columns` and `spacing` from the top level. It will also inherit the props of the top-level grid if it receives those props.

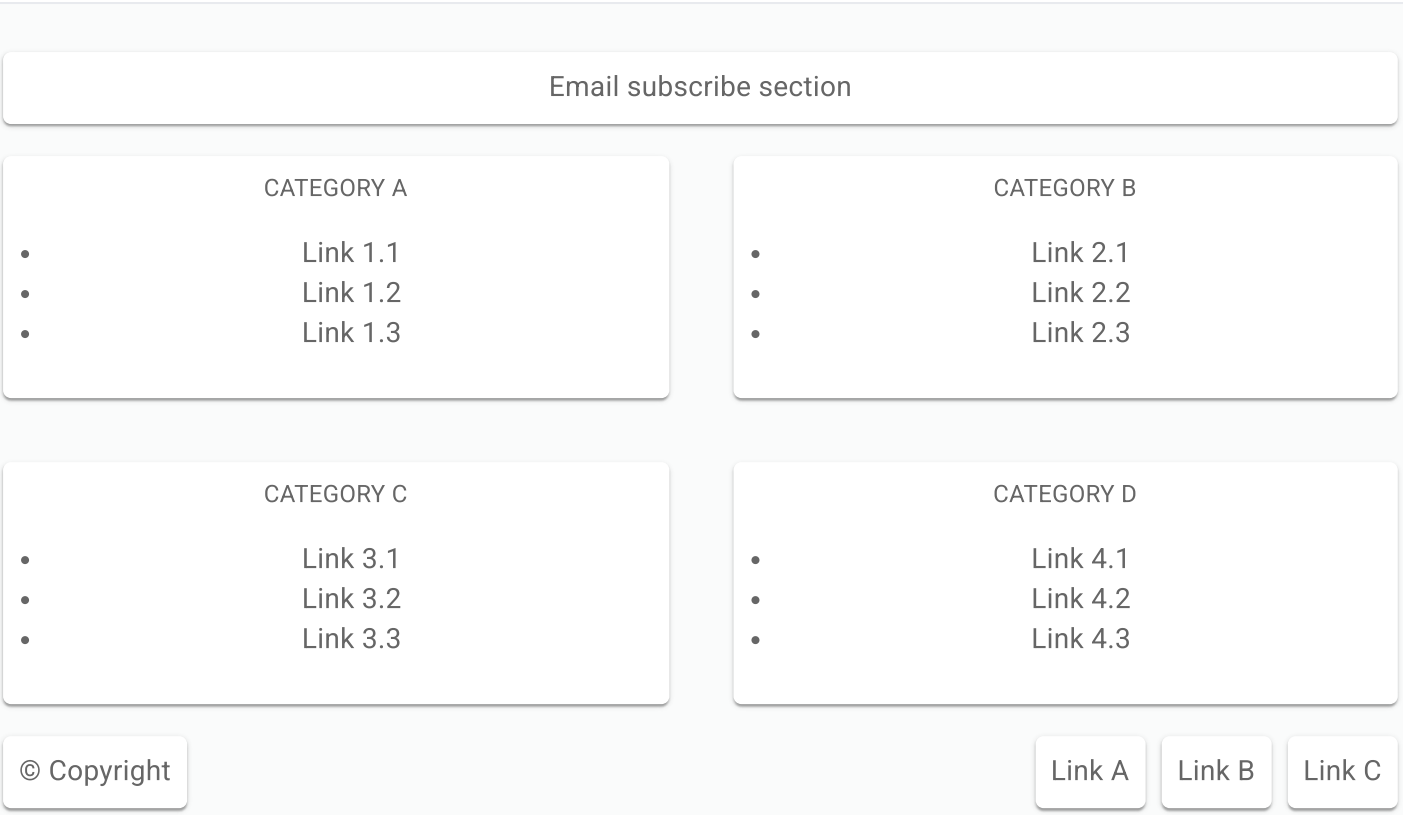
Note that a nested grid container should be a direct child of another grid container. If there are non-grid elements in between, the grid container will start as the new root container.

```jsx
<Grid container>
  <Grid container> // A nested grid container that inherits columns and spacing from above.
    <div>
      <Grid container> // A new root grid container with its own variables scope.
```

Copy

Inheriting spacing

A nested grid container inherits the row and column spacing from its parent unless the `spacing` prop is specified to the instance.



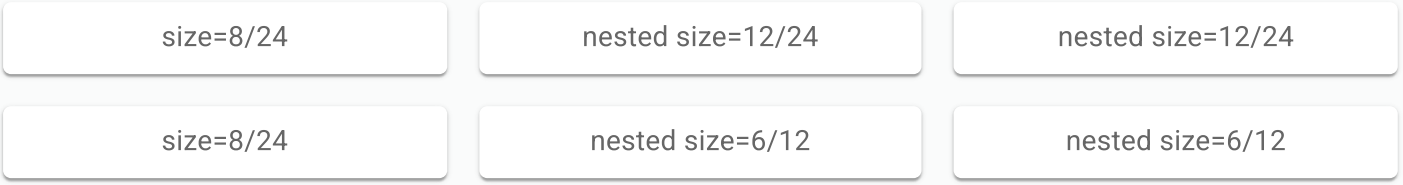
Edit in Chat | JS | TS | Hide code

```jsx
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function NestedGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={2}>
        <Grid size={{ xs: 12, md: 5, lg: 4 }}>
          <Item>Email subscribe section</Item>
        </Grid>
        <Grid container spacing={4} size={{ xs: 12, md: 7, lg: 8 }}>
          <Grid size={{ xs: 6, lg: 3 }}>
            <Item>
              <Box
                id="category-a"
```

```
            sx={{ fontSize: '12px', textTransform: 'uppercase' }}
          >
            Category A
          </Box>
          <Box component="ul" aria-labelledby="category-a" sx={{ pl: 2 }}>
            <li>Link 1.1</li>
            <li>Link 1.2</li>
            <li>Link 1.3</li>
          </Box>
        </Item>
```

Inheriting columns

A nested grid container inherits the columns from its parent unless the `columns` prop is specified to the instance.



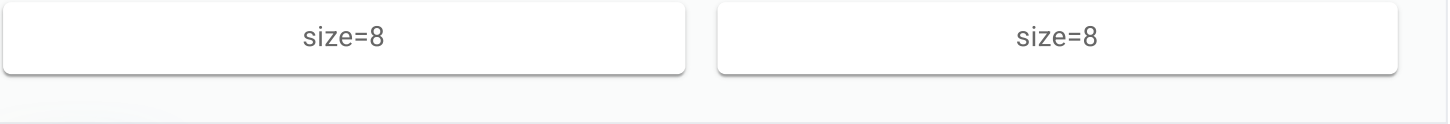
Edit in Chat | JS | TS | Hide code

```
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function NestedGridColumns() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={2} columns={24}>
        <Grid size={8}>
          <Item>size=8/24</Item>
        </Grid>
        <Grid container size={16}>
          <Grid size={12}>
            <Item>nested size=12/24</Item>
          </Grid>
          <Grid size={12}>
```

```
            <Item>nested size=12/24</Item>
          </Grid>
        </Grid>
        <Grid size={8}>
          <Item>size=8/24</Item>
        </Grid>
        <Grid container columns={12} size={16}>
          <Grid size={6}>
            <Item>nested size=6/12</Item>
          </Grid>
```

Columns

Use the `columns` prop to change the default number of columns (12) in the grid, as shown below:



Edit in Chat JS TS Collapse code

```
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function ColumnsGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={2} columns={16}>
        <Grid size={8}>
          <Item>size=8</Item>
        </Grid>
        <Grid size={8}>
          <Item>size=8</Item>
        </Grid>
      </Grid>
    </Box>
```

```
  );
}
```

Offset

The `offset` prop pushes an item to the right side of the grid. This props accepts:

- numbers—for example, `offset={{ md: 2 }}` pushes an item two columns to the right when the viewport size is equal to or greater than the `md` breakpoint.
- `"auto"`—this pushes the item to the far right side of the grid container.

The demo below illustrates how to use the offset props:

Edit in Chat | JS | TS | Collapse code

```tsx
import { styled } from '@mui/material/styles';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function OffsetGrid() {
  return (
    <Grid container spacing={3} sx={{ flexGrow: 1 }}>
      <Grid size={{ xs: 6, md: 2 }} offset={{ xs: 3, md: 0 }}>
        <Item>1</Item>
      </Grid>
      <Grid size={{ xs: 4, md: 2 }} offset={{ md: 'auto' }}>
        <Item>2</Item>
      </Grid>
```

```
      <Grid size={{ xs: 4, md: 2 }} offset={{ xs: 4, md: 0 }}>
        <Item>3</Item>
      </Grid>
      <Grid size={{ xs: 'grow', md: 6 }} offset={{ md: 2 }}>
        <Item>4</Item>
      </Grid>
    </Grid>
  );
}
```

Custom breakpoints

If you specify custom breakpoints in the theme, you can use those names as grid item props in responsive values:

```
import { ThemeProvider, createTheme } from '@mui/material/styles';

function Demo() {
  return (
    <ThemeProvider
      theme={createTheme({
        breakpoints: {
          values: {
            laptop: 1024,
            tablet: 640,
            mobile: 0,
            desktop: 1280,
          },
        },
      })}
    >
      <Grid container spacing={{ mobile: 1, tablet: 2, laptop: 3 }}>
        {Array.from(Array(4)).map((_, index) => (
          <Grid key={index} size={{ mobile: 6, tablet: 4, laptop: 3 }}>
            <div>{index + 1}</div>
```

Copy

Custom breakpoints affect all responsive values.

TypeScript

You have to set module augmentation on the theme breakpoints interface.

```
declare module '@mui/system' {
  interface BreakpointOverrides {
    // Your custom breakpoints
    laptop: true;
    tablet: true;
    mobile: true;
    desktop: true;
```

Copy

```
    // Remove default breakpoints
    xs: false;
    sm: false;
    md: false;
    lg: false;
    xl: false;
  }
}
```

Customization

Centered elements

To center a grid item's content, specify `display="flex"` directly on the item. Then use `justifyContent` and/or `alignItems` to adjust the position of the content, as shown below:



Edit in Chat | JS | TS | Collapse code

```tsx
import Avatar from '@mui/material/Avatar';
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';

export default function CenteredElementGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={2} minHeight={160}>
        <Grid display="flex" justifyContent="center" alignItems="center" size="grow">
          <Avatar src="/static/images/avatar/1.jpg" />
        </Grid>
        <Grid display="flex" justifyContent="center" alignItems="center">
          <Avatar src="/static/images/avatar/2.jpg" />
        </Grid>
        <Grid display="flex" justifyContent="center" alignItems="center" size="grow">
          <Avatar src="/static/images/avatar/3.jpg" />
        </Grid>
      </Grid>
    </Box>
  );
}
```

Using the `container` prop does not work in this situation because the grid container is designed exclusively to wrap grid items. It cannot wrap other elements.

Full border

Edit in Chat | JS | TS | Hide code

```tsx
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';

export default function FullBorderedGrid() {
  return (
    <Box sx={{ flexGrow: 1, p: 2 }}>
      <Grid
        container
        sx={{
          '--Grid-borderWidth': '1px',
          borderTop: 'var(--Grid-borderWidth) solid',
          borderLeft: 'var(--Grid-borderWidth) solid',
          borderColor: 'divider',
          '& > div': {
            borderRight: 'var(--Grid-borderWidth) solid',
            borderBottom: 'var(--Grid-borderWidth) solid',
            borderColor: 'divider',
          },
```

```
        }}
      >
        {[...Array(6)].map((_, index) => (
          <Grid
            key={index}
            minHeight={160}
            size={{
              xs: 12,
              sm: 6,
              md: 4,
              lg: 3,
            }}
          />
        ))}
      </Grid>
    </Box>
  );
}
```

Half border

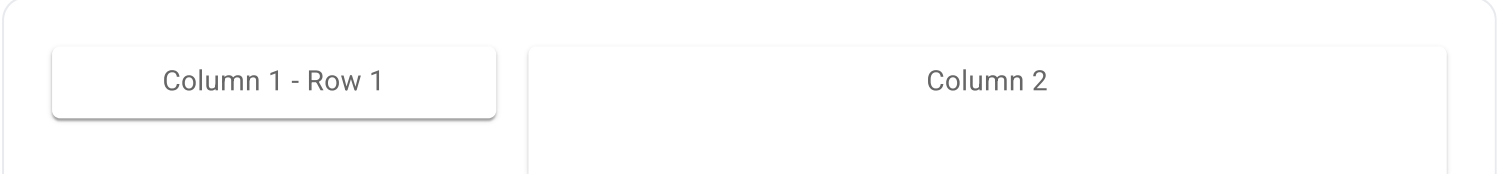
Edit in Chat | JS | TS | Hide code

```tsx
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';

export default function HalfBorderedGrid() {
  const colWidth = { xs: 12, sm: 6, md: 4, lg: 3 } as const;
  return (
    <Box sx={{ flexGrow: 1, p: 2 }}>
      <Grid
        container
        sx={(theme) => ({
          '--Grid-borderWidth': '1px',
          borderTop: 'var(--Grid-borderWidth) solid',
          borderColor: 'divider',
          '& > div': {
            borderRight: 'var(--Grid-borderWidth) solid',
            borderBottom: 'var(--Grid-borderWidth) solid',
            borderColor: 'divider',
            ...(Object.keys(colWidth) as Array<keyof typeof colWidth>).reduce(
              (result, key) => ({
                ...result,
                [`&:nth-of-type(${12 / colWidth[key]}n)`]: {
                  [theme.breakpoints.only(key)]: {
                    borderRight: 'none',
                  },
                },
              }),
              {},
            ),
          },
        })}
      >
        {[...Array(6)].map((_, index) => (
          <Grid key={index} size={colWidth} minHeight={160} />
        ))}
      </Grid>
    </Box>
```

Limitations

Column direction

Using `direction="column"` or `direction="column-reverse"` is not supported. The Grid component is specifically designed to subdivide a layout into columns, not rows. You should not use the Grid component on its own to stack layout elements vertically. Instead, you should use the Stack component inside of a Grid to create vertical layouts as shown below:



Edit in Chat JS TS Collapse code

```tsx
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Paper from '@mui/material/Paper';
import Grid from '@mui/material/Grid';
import Stack from '@mui/material/Stack';

const Item = styled(Paper)(({ theme }) => ({
  backgroundColor: '#fff',
  ...theme.typography.body2,
  padding: theme.spacing(1),
  textAlign: 'center',
  color: (theme.vars ?? theme).palette.text.secondary,
  ...theme.applyStyles('dark', {
    backgroundColor: '#1A2027',
  }),
}));

export default function ColumnLayoutInsideGrid() {
  return (
    <Box sx={{ flexGrow: 1 }}>
      <Grid container spacing={2}>
        <Grid size={4}>
          <Stack spacing={2}>
            <Item>Column 1 - Row 1</Item>
            <Item>Column 1 - Row 2</Item>
            <Item>Column 1 - Row 3</Item>
          </Stack>
        </Grid>
        <Grid size={8}>
          <Item sx={{ height: '100%', boxSizing: 'border-box' }}>Column 2</Item>
        </Grid>
      </Grid>
    </Box>
  );
}
```

API

See the documentation below for a complete reference to all of the props and classes available to the components mentioned here.

- `<Grid />`

- `<PigmentGrid />`

Container

GridLegacy

MUI • Blog • Store