

# Table

Tables display sets of data. They can be fully customized.



**MUI for enterprise** - Save time and reduce risk. Managed open source — backed by maintainers.

ad by MUI

Tables display information in a way that's easy to scan, so that users can look for patterns and insights. They can be embedded in primary content, such as cards. They can include:

- A corresponding visualization
- Navigation
- Tools to query and manipulate data

[View as Markdown](#)[Feedback](#)[Bundle size](#)[Source](#)[W3C WAI-ARIA](#)[Material Design](#)[Figma](#)[Sketch](#)

## Introduction



Tables are implemented using a collection of related components:

- `<TableContainer />`: A wrapper that provides horizontally scrolling behavior for the `<Table />` component.
- `<Table />`: The main component for the table element. Renders as a `<table>` by default.
- `<TableHead />`: The container for the header row(s) of `<Table />`. Renders as a `<thead>` by default.
- `<TableBody />`: The container for the body rows of `<Table />`. Renders as a `<tbody>` by default.
- `<TableRow />`: A row in a table. Can be used in `<TableHead />`, `<TableBody />`, or `<TableFooter />`. Renders as a `<tr>` by default.
- `<TableCell />`: A cell in a table. Can be used in `<TableRow />`. Renders as a `<th>` in `<TableHead />` and `<td>` in `<TableBody />` by default.
- `<TableFooter />`: An optional container for the footer row(s) of the table. Renders as a `<tfoot>` by default.
- `<TablePagination />`: A component that provides controls for paginating table data. See the ['Sorting & selecting' example](#) and ['Custom Table Pagination Action' example](#).

- `<TableSortLabel />`: A component used to display sorting controls for column headers, allowing users to sort data in ascending or descending order. See the ['Sorting & selecting' example](#).

# Basic table

A simple example with no frills.

Dessert (100g serving)	Calories	Fat (g)	Carbs (g)	Protein (g)
Frozen yoghurt	159	6	24	4
Ice cream sandwich	237	9	37	4.3
Eclair	262	16	24	6
Cupcake	305	3.7	67	4.3
Gingerbread	356	16	49	3.9

Edit in Chat

JSTS

Hide code



```
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';

function createData(
  name: string,
  calories: number,
  fat: number,
  carbs: number,
  protein: number,
) {
  return { name, calories, fat, carbs, protein };
}

const rows = [
  createData('Frozen yoghurt', 159, 6.0, 24, 4.0),
  createData('Ice cream sandwich', 237, 9.0, 37, 4.3),
  createData('Eclair', 262, 16.0, 24, 6.0),
  createData('Cupcake', 305, 3.7, 67, 4.3),
  createData('Gingerbread', 356, 16.0, 49, 3.9),
];

export default function BasicTable() {
```

```

return (
  <TableContainer component={Paper}>
    <Table sx={{ minWidth: 650 }} aria-label="simple table">
      <TableHead>
        <TableRow>
          <TableCell>Dessert (100g serving)</TableCell>
          <TableCell align="right">Calories</TableCell>
          <TableCell align="right">Fat (g)</TableCell>
          <TableCell align="right">Carbs (g)</TableCell>
          <TableCell align="right">Protein (g)</TableCell>
        </TableRow>
      </TableHead>
    </Table>
  </TableContainer>
)

```

## Data table

The `Table` component has a close mapping to the native `<table>` elements. This constraint makes building rich data tables challenging.

The [DataGrid component](#) is designed for use-cases that are focused on handling large amounts of tabular data. While it comes with a more rigid structure, in exchange, you gain more powerful features.

<input type="checkbox"/>	ID	First name	Last name	Age	Full name
<input type="checkbox"/>	1	Jon	Snow	35	Jon Snow
<input type="checkbox"/>	2	Cersei	Lannister	42	Cersei Lannister
<input type="checkbox"/>	3	Jaime	Lannister	45	Jaime Lannister
<input type="checkbox"/>	4	Arya	Stark	16	Arya Stark
<input type="checkbox"/>	5	Daenerys	Targaryen		Daenerys Targaryen
<div>             Rows per page: 5             1–5 of 9             &lt;             &gt; </div>					

[Edit in Chat](#)
[JS](#)
[TS](#)
[Collapse code](#)


```

import { DataGrid, GridColDef } from '@mui/x-data-grid';
import Paper from '@mui/material/Paper';

const columns: GridColDef[] = [
  { field: 'id', headerName: 'ID', width: 70 },
  { field: 'firstName', headerName: 'First name', width: 130 },

```

```

    { field: 'lastName', headerName: 'Last name', width: 130 },
    {
      field: 'age',
      headerName: 'Age',
      type: 'number',
      width: 90,
    },
    {
      field: 'fullName',
      headerName: 'Full name',
      description: 'This column has a value getter and is not sortable.',
      sortable: false,
      width: 160,
      valueGetter: (value, row) => `${row.firstName || ''} ${row.lastName || ''}`,
    },
  ],

  const rows = [
    { id: 1, lastName: 'Snow', firstName: 'Jon', age: 35 },
    { id: 2, lastName: 'Lannister', firstName: 'Cersei', age: 42 },
    { id: 3, lastName: 'Lannister', firstName: 'Jaime', age: 45 },
    { id: 4, lastName: 'Stark', firstName: 'Arya', age: 16 },
    { id: 5, lastName: 'Targaryen', firstName: 'Daenerys', age: null },
    { id: 6, lastName: 'Melisandre', firstName: null, age: 150 },
    { id: 7, lastName: 'Clifford', firstName: 'Ferrara', age: 44 },
    { id: 8, lastName: 'Frances', firstName: 'Rossini', age: 36 },
    { id: 9, lastName: 'Roxie', firstName: 'Harvey', age: 65 },
  ];

  const paginationModel = { page: 0, pageSize: 5 };

  export default function DataTable() {

```

## Dense table

A simple example of a dense table with no frills.

Dessert (100g serving)	Calories	Fat (g)	Carbs (g)	Protein (g)
Frozen yoghurt	159	6	24	4
Ice cream sandwich	237	9	37	4.3
Eclair	262	16	24	6
Cupcake	305	3.7	67	4.3
Gingerbread	356	16	49	3.9

[✦ Edit in Chat](#)
[JS](#)
[TS](#)
[Hide code](#)


```

import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';

function createData(
  name: string,
  calories: number,
  fat: number,
  carbs: number,
  protein: number,
) {
  return { name, calories, fat, carbs, protein };
}

const rows = [
  createData('Frozen yoghurt', 159, 6.0, 24, 4.0),
  createData('Ice cream sandwich', 237, 9.0, 37, 4.3),
  createData('Eclair', 262, 16.0, 24, 6.0),
  createData('Cupcake', 305, 3.7, 67, 4.3),
  createData('Gingerbread', 356, 16.0, 49, 3.9),
];

export default function DenseTable() {
  return (
    <TableContainer component={Paper}>
      <Table sx={{ minWidth: 650 }} size="small" aria-label="a dense table">
        <TableHead>
          <TableRow>
            <TableCell>Dessert (100g serving)</TableCell>
            <TableCell align="right">Calories</TableCell>
            <TableCell align="right">Fat (g)</TableCell>
            <TableCell align="right">Carbs (g)</TableCell>
            <TableCell align="right">Protein (g)</TableCell>
          </TableRow>
        </TableHead>
        <TableBody>
          {rows.map((row) => (
            <TableRow>
              <TableCell>{row.name}</TableCell>
              <TableCell align="right">{row.calories}</TableCell>
              <TableCell align="right">{row.fat}</TableCell>
              <TableCell align="right">{row.carbs}</TableCell>
              <TableCell align="right">{row.protein}</TableCell>
            </TableRow>
          ))}
        </TableBody>
      </Table>
    </TableContainer>
  );
}

```

## Sorting & selecting



This example demonstrates the use of `Checkbox` and clickable rows for selection, with a custom `Toolbar`. It uses the `TableSortLabel` component to help style column headings.

The Table has been given a fixed width to demonstrate horizontal scrolling. In order to prevent the pagination controls from scrolling, the `TablePagination` component is used outside of the Table. (The ['Custom Table Pagination Action' example](#) below shows the pagination within the `TableFooter`.)

Nutrition



<input type="checkbox"/> Dessert (100g serving)	↑ Calories	Fat (g)	Carbs (g)	Protein (g)
<input type="checkbox"/> Frozen yoghurt	159	6	24	10
<input type="checkbox"/> Ice cream sandwich	237	9	37	5
<input type="checkbox"/> Eclair	262	16	24	4
<input type="checkbox"/> Cupcake	305	3.7	67	4
<input type="checkbox"/> Marshmallow	318	0	81	3
Rows per page: 5 ▾ 1–5 of 13 < >				

☐ Dense padding

Edit in Chat

JSTS

Hide code



```
import * as React from 'react';
import { alpha } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TablePagination from '@mui/material/TablePagination';
import TableRow from '@mui/material/TableRow';
import TableSortLabel from '@mui/material/TableSortLabel';
import Toolbar from '@mui/material/Toolbar';
import Typography from '@mui/material/Typography';
import Paper from '@mui/material/Paper';
import Checkbox from '@mui/material/Checkbox';
import IconButton from '@mui/material/IconButton';
import Tooltip from '@mui/material/Tooltip';
import FormControlLabel from '@mui/material/FormControlLabel';
import Switch from '@mui/material/Switch';
import DeleteIcon from '@mui/icons-material/Delete';
import FilterListIcon from '@mui/icons-material/FilterList';
import { visuallyHidden } from '@mui/utils';

interface Data {
  id: number;
  calories: number;
  carbs: number;
  fat: number;
  name: string;
  protein: number;
}
```

```
function createData(  
  id: number,  
  name: string,  
  calories: number,  
  fat: number,  
  carbs: number,
```

# Customization



Here is an example of customizing the component. You can learn more about this in the [overrides documentation page](#).

Dessert (100g serving)	Calories	Fat (g)	Carbs (g)	Protein (g)
Frozen yoghurt	159	6	24	4
Ice cream sandwich	237	9	37	4.3
Eclair	262	16	24	6
Cupcake	305	3.7	67	4.3
Gingerbread	356	16	49	3.9

Edit in Chat

JS

TS

Hide code

```
import { styled } from '@mui/material/styles';  
import Table from '@mui/material/Table';  
import TableBody from '@mui/material/TableBody';  
import TableCell, { tableCellClasses } from '@mui/material/TableCell';  
import TableContainer from '@mui/material/TableContainer';  
import TableHead from '@mui/material/TableHead';  
import TableRow from '@mui/material/TableRow';  
import Paper from '@mui/material/Paper';  
  
const StyledTableCell = styled(TableCell)(({ theme }) => ({  
  [`&.${tableCellClasses.head}`]: {  
    backgroundColor: theme.palette.common.black,  
    color: theme.palette.common.white,  
  },  
  [`&.${tableCellClasses.body}`]: {  
    fontSize: 14,  
  },  
});
```

```

    }));

const StyledTableRow = styled(TableRow)(({ theme }) => ({
  '&:nth-of-type(odd)': {
    backgroundColor: theme.palette.action.hover,
  },
  // hide last border
  '&:last-child td, &:last-child th': {
    border: 0,
  },
}));

function createData(
  name: string,
  calories: number,
  fat: number,
  carbs: number,
  protein: number,
) {
  return { name, calories, fat, carbs, protein };
}

```

## Custom pagination options

It's possible to customize the options shown in the "Rows per page" select using the `rowsPerPageOptions` prop. You should either provide an array of:

- **numbers**, each number will be used for the option's label and value.

```
<TablePagination rowsPerPageOptions={[10, 50]} />
```

Copy

- **objects**, the `value` and `label` keys will be used respectively for the value and label of the option (useful for language strings such as 'All').

```
<TablePagination rowsPerPageOptions={[10, 50, { value: -1, label: 'All' }]} />
```

Copy

## Custom pagination actions

The `ActionsComponent` prop of the `TablePagination` component allows the implementation of custom actions.

Frozen yoghurt	159	6
Ice cream sandwich	237	9
<div>             Rows per page: 5 ▾ 1–5 of 13              &lt; &lt; &gt; &gt;            </div>		

Eclair	262	16
Cupcake	305	3.7
Marshmallow	318	0
Rows per page: ▼ 1–5 of 13		

[Edit in Chat](#)
[JS](#)
[TS](#)
[Hide code](#)


```
import * as React from 'react';
import { useTheme } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableFooter from '@mui/material/TableFooter';
import TablePagination from '@mui/material/TablePagination';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import IconButton from '@mui/material/IconButton';
import FirstPageIcon from '@mui/icons-material/FirstPage';
import KeyboardArrowLeft from '@mui/icons-material/KeyboardArrowLeft';
import KeyboardArrowRight from '@mui/icons-material/KeyboardArrowRight';
import LastPageIcon from '@mui/icons-material/LastPage';

interface TablePaginationActionsProps {
  count: number;
  page: number;
  rowsPerPage: number;
  onPageChange: (
    event: React.MouseEvent<HTMLButtonElement>,
    newPage: number,
  ) => void;
}

function TablePaginationActions(props: TablePaginationActionsProps) {
  const theme = useTheme();
  const { count, page, rowsPerPage, onPageChange } = props;

  const handleFirstPageButtonClick = (
    event: React.MouseEvent<HTMLButtonElement>,
  ) => {
    onPageChange(event, 0);
  };

  const handleBackButtonClick = (event: React.MouseEvent<HTMLButtonElement>) => {
```

# Sticky header+

Here is an example of a table with scrollable rows and fixed column headers. It leverages the `stickyHeader` prop.

Name	ISO Code	Population	Size (km²)
India	IN	1,324,171,354	3,287,263
China	CN	1,403,500,365	9,596,961
Italy	IT	60,483,973	301,340
United States	US	327,167,434	9,833,520
Canada	CA	37,602,103	9,984,670
Australia	AU	25,475,400	7,692,024
Germany	DE	83,019,200	357,578
Rows per page: 10 ▾ 1–10 of 15 < >			

Edit in Chat

JS

TS

Hide code

```
import * as React from 'react';
import Paper from '@mui/material/Paper';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TablePagination from '@mui/material/TablePagination';
import TableRow from '@mui/material/TableRow';

interface Column {
  id: 'name' | 'code' | 'population' | 'size' | 'density';
  label: string;
  minWidth?: number;
  align?: 'right';
  format?: (value: number) => string;
}

const columns: readonly Column[] = [
  { id: 'name', label: 'Name', minWidth: 170 },
```

```
{ id: 'code', label: 'ISO\u00a0Code', minWidth: 100 },
{
  id: 'population',
  label: 'Population',
  minWidth: 170,
  align: 'right',
  format: (value: number) => value.toLocaleString('en-US'),
},
{
  id: 'size',
  label: 'Size\u00a0(km\u00b2)',
  minWidth: 170,
  align: 'right',
  format: (value: number) => value.toLocaleString('en-US'),
},
{
  id: 'density',
```

# Column grouping



You can group column headers by rendering multiple table rows inside a table head:

```
<TableHead>
  <TableRow />
  <TableRow />
</TableHead>
```

Copy

Country		Details	
Name	ISO Code	Population	Size (km²)
India	IN	1,324,171,354	3,287,263
China	CN	1,403,500,365	9,596,961
Italy	IT	60,483,973	301,340
United States	US	327,167,434	9,833,520
Canada	CA	37,602,103	9,984,670
Australia	AU	25,475,400	7,692,024

[✈ Edit in Chat](#)

JS

TS

[Hide code](#)

```
import * as React from 'react';
import Paper from '@mui/material/Paper';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TablePagination from '@mui/material/TablePagination';
import TableRow from '@mui/material/TableRow';
```

```
interface Column {
  id: 'name' | 'code' | 'population' | 'size' | 'density';
  label: string;
  minWidth?: number;
  align?: 'right';
  format?: (value: number) => string;
}
```

```
const columns: Column[] = [
  { id: 'name', label: 'Name', minWidth: 170 },
  { id: 'code', label: 'ISO\u00a0Code', minWidth: 100 },
  {
    id: 'population',
    label: 'Population',
    minWidth: 170,
    align: 'right',
    format: (value: number) => value.toLocaleString('en-US'),
  },
  {
    id: 'size',
    label: 'Size\u00a0(km\u00b2)',
    minWidth: 170,
    align: 'right',
    format: (value: number) => value.toLocaleString('en-US'),
  },
  {
    id: 'density',
    label: 'Density',
```

## Collapsible table



An example of a table with expandable rows, revealing more information. It utilizes the [Collapse](#) component.

	Dessert (100g serving)	Calories	Fat (g)	Carbs (g)	Protein (g)
▼	Frozen yoghurt	159	6	24	4
▼	Ice cream sandwich	237	9	37	4.3
▼	Eclair	262	16	24	6
▼	Cupcake	305	3.7	67	4.3
▼	Gingerbread	356	16	49	3.9

 Edit in Chat

JS

TS

Hide code

     

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Collapse from '@mui/material/Collapse';
import IconButton from '@mui/material/IconButton';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Typography from '@mui/material/Typography';
import Paper from '@mui/material/Paper';
import KeyboardArrowDownIcon from '@mui/icons-material/KeyboardArrowDown';
import KeyboardArrowUpIcon from '@mui/icons-material/KeyboardArrowUp';

function createData(
  name: string,
  calories: number,
  fat: number,
  carbs: number,
  protein: number,
  price: number,
) {
  return {
    name,
    calories,
    fat,
    carbs,
    protein,
    price,
    history: [
```

```
{
  date: '2020-01-05',
  customerId: '11091700',
  amount: 3,
},
{
  date: '2020-01-03'
```

# Spanning table

A simple example with spanning rows & columns.

Details			Price
Desc	Qty.	Unit	Sum
Paperclips (Box)	100	1.15	115.00
Paper (Case)	10	45.99	459.90
Waste Basket	2	17.99	35.98
Subtotal			610.88
Tax			7 % 42.76
Total			653.64

```
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';

const TAX_RATE = 0.07;

function ccyFormat(num: number) {
  return ` ${num.toFixed(2)} `;
}
```

```

function priceRow(qty: number, unit: number) {
  return qty * unit;
}

function createRow(desc: string, qty: number, unit: number) {
  const price = priceRow(qty, unit);
  return { desc, qty, unit, price };
}

interface Row {
  desc: string;
  qty: number;
  unit: number;
  price: number;
}

function subtotal(items: readonly Row[]) {
  return items.map(({ price }) => price).reduce((sum, i) => sum + i, 0);
}

const rows = [
  createRow('Paperclips (Box)', 100, 1.15),
  createRow('Paper (Case)', 10, 45.99),
  createRow('Waste Basket', 2, 17.99),

```

## Virtualized table



In the following example, we demonstrate how to use [react-virtuoso](#) with the `Table` component. It renders 200 rows and can easily handle more. Virtualization helps with performance issues.

First Name	Last Name	Age	State	Phone Number
Seth	Lucas	26	Colorado	(775) 614-2403
Todd	Kondo	34	Oklahoma	(617) 700-9729
Ophelia	Becucci	18	Alabama	(381) 636-5722
David	Colzi	43	Hawaii	(631) 229-4297
Brian	Browne	47	Texas	(335) 954-1418
Alberta	Patrick	39	Maine	(209) 995-8275
Michael	Herrera	20	Tennessee	(724) 716-5456

[✈ Edit in Chat](#)

JS

TS

[Collapse code](#)

```
import * as React from 'react';
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';
import { TableVirtuoso, TableComponents } from 'react-virtuoso';
import Chance from 'chance';
```

```
interface Data {
  id: number;
  firstName: string;
  lastName: string;
  age: number;
  phone: string;
  state: string;
}
```

```
interface ColumnData {
  dataKey: keyof Data;
  label: string;
  numeric?: boolean;
  width?: number;
}
```

```
const chance = new Chance(42);
```

```
function createData(id: number): Data {
  return {
```

```
id,
firstName: chance.first(),
lastName: chance.last(),
age: chance.age(),
phone: chance.phone(),
state: chance.state({ full: true }),
};
```

## Accessibility

(WAI tutorial: <https://www.w3.org/WAI/tutorials/tables/>)

## Caption

A caption functions like a heading for a table. Most screen readers announce the content of captions. Captions help users to find a table and understand what it's about and decide if they want to read it.

Dessert (100g serving)	Calories	Fat (g)	Carbs (g)	Protein (g)
Frozen yoghurt	159	6	24	4
Ice cream sandwich	237	9	37	4.3
Eclair	262	16	24	6
A basic table example with a caption				

Edit in Chat

JSTS

Hide code



```
import Table from '@mui/material/Table';
import TableBody from '@mui/material/TableBody';
import TableCell from '@mui/material/TableCell';
import TableContainer from '@mui/material/TableContainer';
import TableHead from '@mui/material/TableHead';
import TableRow from '@mui/material/TableRow';
import Paper from '@mui/material/Paper';

function createData(
  name: string,
  calories: number,
  fat: number,
  carbs: number,
  protein: number,
) {
  return { name, calories, fat, carbs, protein };
}
```

```

}

const rows = [
  createData('Frozen yoghurt', 159, 6.0, 24, 4.0),
  createData('Ice cream sandwich', 237, 9.0, 37, 4.3),
  createData('Eclair', 262, 16.0, 24, 6.0),
];

export default function AccessibleTable() {
  return (
    <TableContainer component={Paper}>
      <Table sx={{ minWidth: 650 }} aria-label="caption table">
        <caption>A basic table example with a caption</caption>
        <TableHead>
          <TableRow>
            <TableCell>Dessert (100g serving)</TableCell>
            <TableCell align="right">Calories</TableCell>
            <TableCell align="right">Fat (g)</TableCell>
            <TableCell align="right">Carbs (g)</TableCell>
            <TableCell align="right">Protein (g)</TableCell>
          </TableRow>
        </TableHead>

```

## API



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

- [<Table />](#)
- [<TableBody />](#)
- [<TableCell />](#)
- [<TableContainer />](#)
- [<TableFooter />](#)
- [<TableHead />](#)
- [<TablePagination />](#)
- [<TableRow />](#)
- [<TableSortLabel />](#)

Edit this page

Was this page helpful?

[< List](#)

[Tooltip >](#)