Table

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

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

{{"component": "modules/components/ComponentLinkHeader.js"}}

Basic table

A simple example with no frills.

{{"demo": "BasicTable.js", "bg": true}}

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.

{{"demo": "DataTable.js", "bg": "inline"}}

Dense table

A simple example of a dense table with no frills.

{{"demo": "DenseTable.js", "bg": true}}

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.)

{{"demo": "EnhancedTable.js", "bg": true}}

Customization

Here is an example of customizing the component. You can learn more about this in the overrides documentation page.

{{"demo": "CustomizedTables.js", "bg": true}}

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]} />
```

- **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' }]}
/>
```

Custom pagination actions

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

{{"demo": "CustomPaginationActionsTable.js", "bg": true}}

Sticky header

Here is an example of a table with scrollable rows and fixed column headers. It leverages the `stickyHeader` prop. (⚠ no IE 11 support)

{{"demo": "StickyHeadTable.js", "bg": true}}

Column grouping

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

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

{{"demo": "ColumnGroupingTable.js", "bg": true}}

Collapsible table

An example of a table with expandable rows, revealing more information. It utilizes the `Collapse` component.

{{"demo": "CollapsibleTable.js", "bg": true}}

Spanning table

A simple example with spanning rows & columns.

{{"demo": "SpanningTable.js", "bg": true}}

Virtualized table

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

{{"demo": "ReactVirtualizedTable.js", "bg": 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.

{{"demo": "AcccessibleTable.js", "bg": true}}

Unstyled

If you would like to use an unstyled Table, you can use the primitive HTML elements and enhance the table with the TablePaginationUnstyled component. See the demos in the unstyled table pagination docs