Pagination

The Pagination component enables the user to select a specific page from a range of pages.

View as Markdown | Feedback | Bundle size | Source | Figma | Sketch

Basic pagination

Edit in Chat | JS | TS | Collapse code

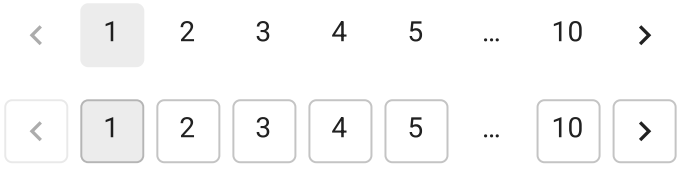
```
import Pagination from '@mui/material/Pagination';
import Stack from '@mui/material/Stack';

export default function BasicPagination() {
  return (
    <Stack spacing={2}>
      <Pagination count={10} />
      <Pagination count={10} color="primary" />
      <Pagination count={10} color="secondary" />
      <Pagination count={10} disabled />
    </Stack>
  );
}
```

Outlined pagination

Edit in Chat | JS | TS | Collapse code

```
import Pagination from '@mui/material/Pagination';
import Stack from '@mui/material/Stack';

export default function PaginationOutlined() {
  return (
    <Stack spacing={2}>
      <Pagination count={10} variant="outlined" />
      <Pagination count={10} variant="outlined" color="primary" />
      <Pagination count={10} variant="outlined" color="secondary" />
      <Pagination count={10} variant="outlined" disabled />
    </Stack>
  );
}
```

Rounded pagination



Edit in Chat | JS | TS | Collapse code

```
import Pagination from '@mui/material/Pagination';
import Stack from '@mui/material/Stack';

export default function PaginationRounded() {
  return (
    <Stack spacing={2}>
      <Pagination count={10} shape="rounded" />
      <Pagination count={10} variant="outlined" shape="rounded" />
    </Stack>
```

```
  );
}
```

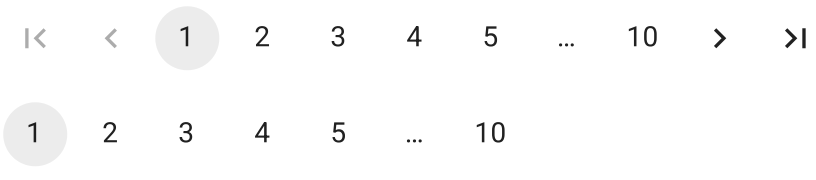
Pagination size

Edit in Chat | JS | TS | Collapse code

```tsx
import Pagination from '@mui/material/Pagination';
import Stack from '@mui/material/Stack';

export default function PaginationSize() {
  return (
    <Stack spacing={2}>
      <Pagination count={10} size="small" />
      <Pagination count={10} />
      <Pagination count={10} size="large" />
    </Stack>
  );
}
```

Buttons

You can optionally enable first-page and last-page buttons, or disable the previous-page and next-page buttons.



Edit in Chat | JS | TS | Collapse code

```tsx
import Pagination from '@mui/material/Pagination';
import Stack from '@mui/material/Stack';
```

```tsx
export default function PaginationButtons() {
  return (
    <Stack spacing={2}>
      <Pagination count={10} showFirstButton showLastButton />
      <Pagination count={10} hidePrevButton hideNextButton />
    </Stack>
  );
}
```

Custom icons

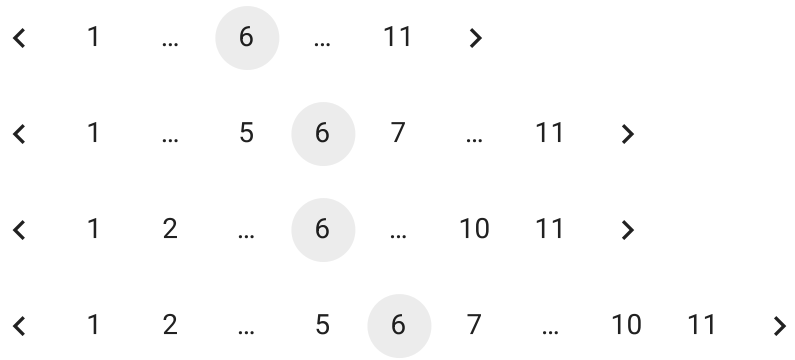
It's possible to customize the control icons.

Edit in Chat | JS | TS | Collapse code

```tsx
import Pagination from '@mui/material/Pagination';
import PaginationItem from '@mui/material/PaginationItem';
import Stack from '@mui/material/Stack';
import ArrowBackIcon from '@mui/icons-material/ArrowBack';
import ArrowForwardIcon from '@mui/icons-material/ArrowForward';

export default function CustomIcons() {
  return (
    <Stack spacing={2}>
      <Pagination
        count={10}
        renderItem={(item) => (
          <PaginationItem
            slots={{ previous: ArrowBackIcon, next: ArrowForwardIcon }}
            {...item}
          />
        )}
      />
    </Stack>
  );
}
```

Pagination ranges

You can specify how many digits to display either side of current page with the `siblingCount` prop, and adjacent to the start and end page number with the `boundaryCount` prop.



Edit in Chat JS TS Collapse code

```tsx
import Pagination from '@mui/material/Pagination';
import Stack from '@mui/material/Stack';

export default function PaginationRanges() {
  return (
    <Stack spacing={2}>
      <Pagination count={11} defaultPage={6} siblingCount={0} />
      <Pagination count={11} defaultPage={6} /> {/* Default ranges */}
      <Pagination count={11} defaultPage={6} siblingCount={0} boundaryCount={2} />
      <Pagination count={11} defaultPage={6} boundaryCount={2} />
    </Stack>
  );
}
```

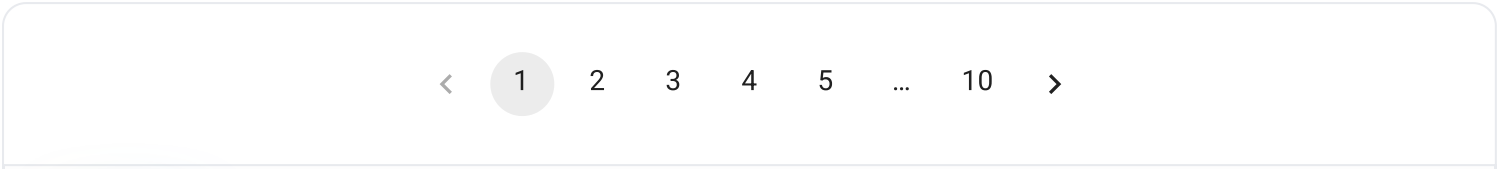
Controlled pagination

Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Typography from '@mui/material/Typography';
import Pagination from '@mui/material/Pagination';
import Stack from '@mui/material/Stack';

export default function PaginationControlled() {
  const [page, setPage] = React.useState(1);
  const handleChange = (event: React.ChangeEvent<unknown>, value: number) => {
    setPage(value);
```

```
  };

  return (
    <Stack spacing={2}>
      <Typography>Page: {page}</Typography>
      <Pagination count={10} page={page} onChange={handleChange} />
    </Stack>
  );
}
```

Router integration



Edit in Chat | JS | TS | Collapse code

```
import { Link, MemoryRouter, Route, Routes, useLocation } from 'react-router';
import Pagination from '@mui/material/Pagination';
import PaginationItem from '@mui/material/PaginationItem';

function Content() {
  const location = useLocation();
  const query = new URLSearchParams(location.search);
  const page = parseInt(query.get('page') || '1', 10);
  return (
    <Pagination
      page={page}
      count={10}
      renderItem={(item) => (
        <PaginationItem
          component={Link}
          to={`/inbox${item.page === 1 ? '' : `?page=${item.page}`}`}
          {...item}
        />
      )}
    />
  );
}

export default function PaginationLink() {
  return (
    <MemoryRouter initialEntries={['/inbox']} initialIndex={0}>
      <Routes>
        <Route path="*" element={<Content />} />
      </Routes>
    </MemoryRouter>
  );
}
```

usePagination

For advanced customization use cases, a headless `usePagination()` hook is exposed. It accepts almost the same options as the Pagination component minus all the props related to the rendering of JSX. The Pagination component is built on this hook.

```
import usePagination from '@mui/material/usePagination';
```

Copy

Edit in Chat | JS | TS | Hide code

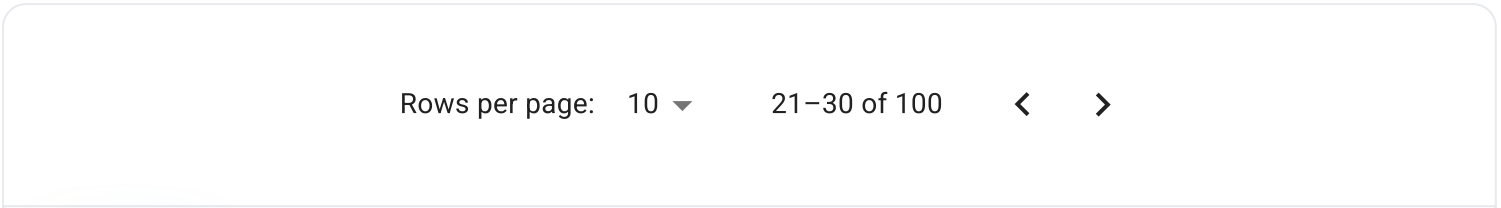
```jsx
import usePagination from '@mui/material/usePagination';
import { styled } from '@mui/material/styles';

const List = styled('ul')({
  listStyle: 'none',
  padding: 0,
  margin: 0,
  display: 'flex',
});

export default function UsePagination() {
  const { items } = usePagination({
    count: 10,
  });

  return (
    <nav>
      <List>
        {items.map(({ page, type, selected, ...item }, index) => {
          let children = null;

          if (type === 'start-ellipsis' || type === 'end-ellipsis') {
            children = '…';
          } else if (type === 'page') {
            children = (
              <button
                type="button"
                style={{
                  fontWeight: selected ? 'bold' : undefined,
                }}
                {...item}
              >
                {page}
              </button>
            );
```

```
        } else {
          children = (
```

Table pagination

The `Pagination` component was designed to paginate a list of arbitrary items when infinite loading isn't used. It's preferred in contexts where SEO is important, for instance, a blog.

For the pagination of a large set of tabular data, you should use the `TablePagination` component.



Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import TablePagination from '@mui/material/TablePagination';

export default function TablePaginationDemo() {
  const [page, setPage] = React.useState(2);
  const [rowsPerPage, setRowsPerPage] = React.useState(10);

  const handleChangePage = (
    event: React.MouseEvent<HTMLButtonElement> | null,
    newPage: number,
  ) => {
    setPage(newPage);
  };

  const handleChangeRowsPerPage = (
    event: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement>,
  ) => {
    setRowsPerPage(parseInt(event.target.value, 10));
    setPage(0);
  };

  return (
    <TablePagination
      component="div"
      count={100}
      page={page}
      onPageChange={handleChangePage}
      rowsPerPage={rowsPerPage}
      onRowsPerPageChange={handleChangeRowsPerPage}
    />
  );
}
```

⚠ Note that the `Pagination` page prop starts at 1 to match the requirement of including the value in the URL, while the `TablePagination` page prop starts at 0 to match the requirement of zero-based JavaScript arrays that come with rendering a lot of tabular data.

You can learn more about this use case in the table section of the documentation.

Accessibility

ARIA

The root node has a role of "navigation" and aria-label "pagination navigation" by default. The page items have an aria-label that identifies the purpose of the item ("go to first page", "go to previous page", "go to page 1" etc.). You can override these using the `getItemAriaLabel` prop.

Keyboard

The pagination items are in tab order, with a tabindex of "0".

API

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

- `<Pagination />`
- `<PaginationItem />`
- `<TablePagination />`
- `<TablePaginationActions />`