Tabs

Tabs make it easy to explore and switch between different views.

Tabs organize and allow navigation between groups of content that are related and at the same level of hierarchy.

View as Markdown | Feedback | Bundle size | Source | WAI-ARIA | Material Design | Figma | Sketch

Introduction

Tabs are implemented using a collection of related components:

- `<Tab />` - the tab element itself. Clicking on a tab displays its corresponding panel.
- `<Tabs />` - the container that houses the tabs. Responsible for handling focus and keyboard navigation between tabs.

ITEM ONE ITEM TWO ITEM THREE

Item One

Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';

interface TabPanelProps {
  children?: React.ReactNode;
  index: number;
  value: number;
```

```
}

function CustomTabPanel(props: TabPanelProps) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`simple-tabpanel-${index}`}
      aria-labelledby={`simple-tab-${index}`}
      {...other}
    >
      {value === index && <Box sx={{ p: 3 }}>{children}</Box>}
    </div>
  );
}

function a11yProps(index: number) {
  return {
    id: `simple-tab-${index}`,
    'aria-controls': `simple-tabpanel-${index}`,
  };
}

export default function BasicTabs() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
```

Basics

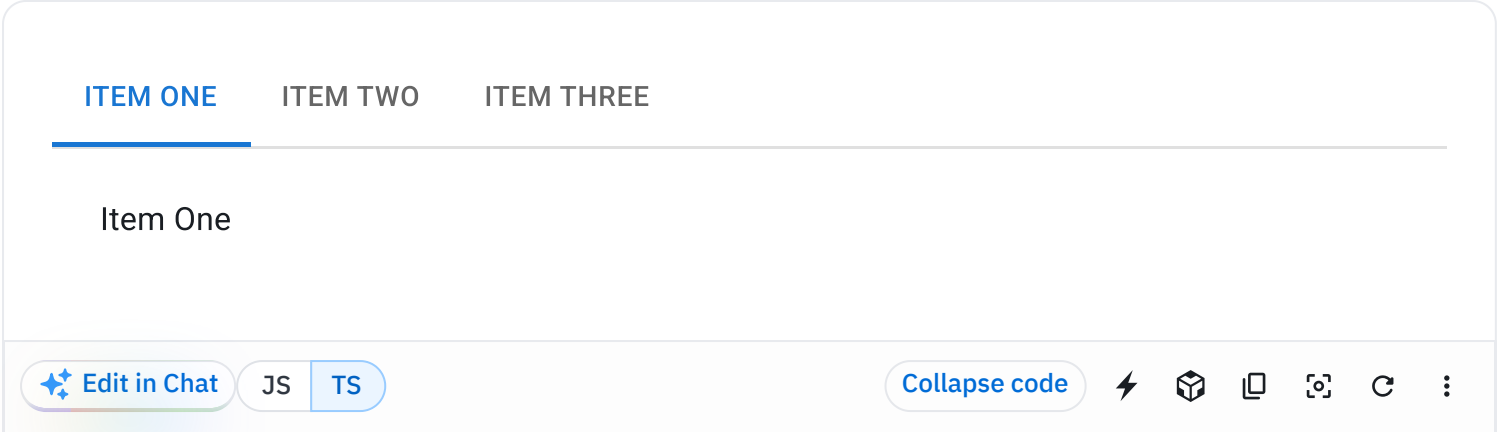
```
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
```

Copy

Experimental API

`@mui/lab` offers utility components that inject props to implement accessible tabs following WAI-ARIA Authoring Practices ↗:

- `<TabList />` - the container that houses the tabs. Responsible for handling focus and keyboard navigation between tabs.
- `<TabPanel />` - the card that hosts the content associated with a tab.
- `<TabContext />` - the top-level component that wraps the Tab List and Tab Panel components.

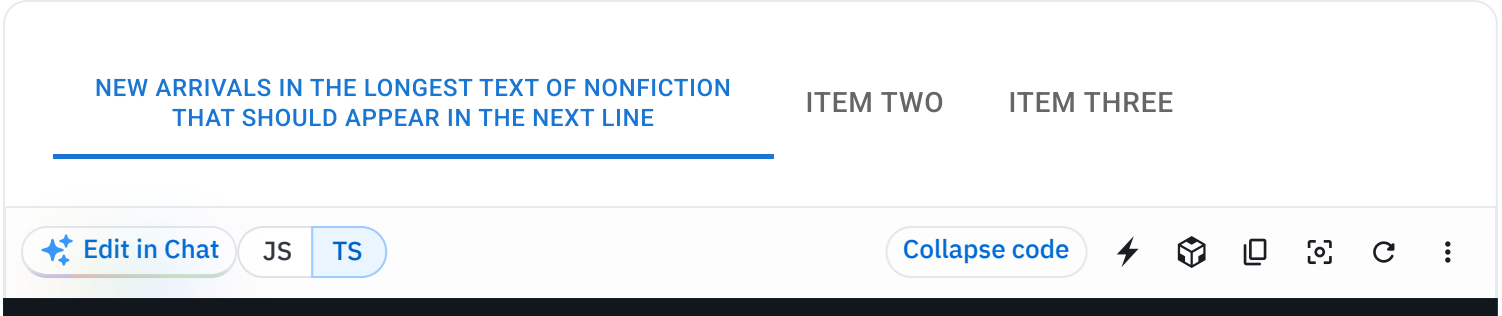


Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Box from '@mui/material/Box';
import Tab from '@mui/material/Tab';
import TabContext from '@mui/lab/TabContext';
import TabList from '@mui/lab/TabList';
import TabPanel from '@mui/lab/TabPanel';

export default function LabTabs() {
  const [value, setValue] = React.useState('1');

  const handleChange = (event: React.SyntheticEvent, newValue: string) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: '100%', typography: 'body1' }}>
      <TabContext value={value}>
        <Box sx={{ borderBottom: 1, borderColor: 'divider' }}>
          <TabList onChange={handleChange} aria-label="lab API tabs example">
            <Tab label="Item One" value="1" />
            <Tab label="Item Two" value="2" />
            <Tab label="Item Three" value="3" />
          </TabList>
        </Box>
        <TabPanel value="1">Item One</TabPanel>
        <TabPanel value="2">Item Two</TabPanel>
        <TabPanel value="3">Item Three</TabPanel>
      </TabContext>
    </Box>
  );
}
```

Wrapped labels

Long labels will automatically wrap on tabs. If the label is too long for the tab, it will overflow, and the text will not be visible.



Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';

export default function TabsWrappedLabel() {
  const [value, setValue] = React.useState('one');

  const handleChange = (event: React.SyntheticEvent, newValue: string) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: '100%' }}>
      <Tabs
        value={value}
        onChange={handleChange}
        aria-label="wrapped label tabs example"
      >
        <Tab
          value="one"
          label="New Arrivals in the Longest Text of Nonfiction that should appear in the next line"
          wrapped
        />
        <Tab value="two" label="Item Two" />
        <Tab value="three" label="Item Three" />
      </Tabs>
    </Box>
  );
}
```

Colored tab

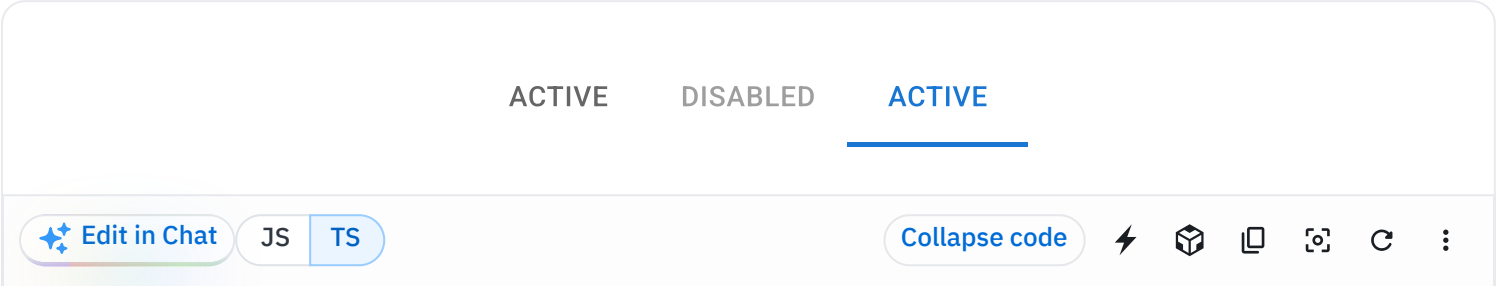
Edit in Chat JS TS Collapse code

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

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

export default function ColorTabs() {
  const [value, setValue] = React.useState('one');

  const handleChange = (event: React.SyntheticEvent, newValue: string) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: '100%' }}>
      <Tabs
        value={value}
        onChange={handleChange}
        textColor="secondary"
        indicatorColor="secondary"
        aria-label="secondary tabs example"
      >
        <Tab value="one" label="Item One" />
        <Tab value="two" label="Item Two" />
        <Tab value="three" label="Item Three" />
      </Tabs>
    </Box>
  );
}
```

Disabled tab

A tab can be disabled by setting the `disabled` prop.



Edit in Chat JS TS Collapse code

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

export default function DisabledTabs() {
  const [value, setValue] = React.useState(2);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Tabs value={value} onChange={handleChange} aria-label="disabled tabs example">
      <Tab label="Active" />
```

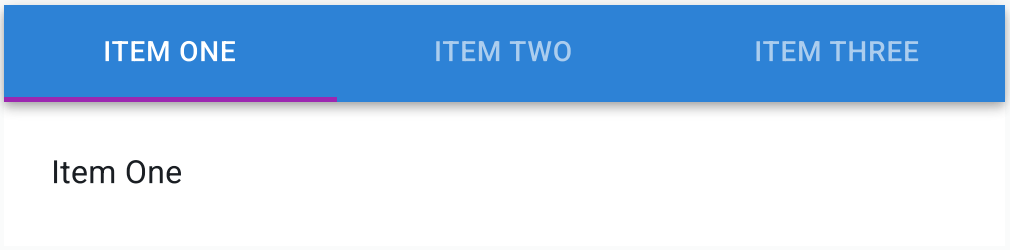
```
      <Tab label="Disabled" disabled />
      <Tab label="Active" />
    </Tabs>
  );
}
```

Fixed tabs

Fixed tabs should be used with a limited number of tabs, and when a consistent placement will aid muscle memory.

Full width

The `variant="fullWidth"` prop should be used for smaller views.



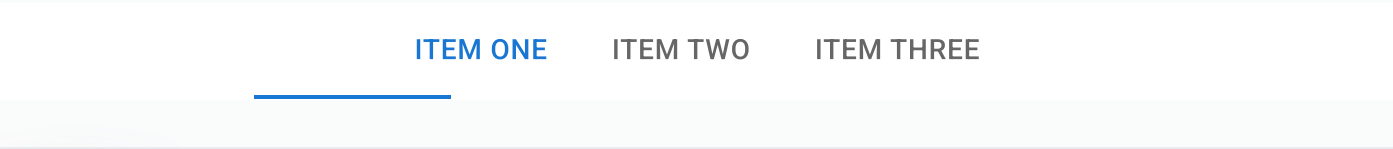
Edit in Chat | JS | TS | Hide code

```tsx
import * as React from 'react';
import { useTheme } from '@mui/material/styles';
import AppBar from '@mui/material/AppBar';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';

interface TabPanelProps {
  children?: React.ReactNode;
  dir?: string;
  index: number;
  value: number;
}

function TabPanel(props: TabPanelProps) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`full-width-tabpanel-${index}`}
      aria-labelledby={`full-width-tab-${index}`}
      {...other}
```

```
    >
      {value === index && (
        <Box sx={{ p: 3 }}>
          <Typography>{children}</Typography>
        </Box>
      )}
    </div>
  );
}

function a11yProps(index: number) {
  return {
    id: `full-width-tab-${index}`,
```

Centered

The `centered` prop should be used for larger views.



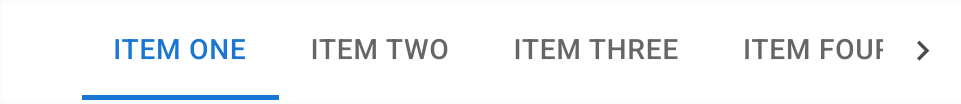
Edit in Chat | JS | TS | Collapse code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';

export default function CenteredTabs() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: '100%', bgcolor: 'background.paper' }}>
      <Tabs value={value} onChange={handleChange} centered>
        <Tab label="Item One" />
        <Tab label="Item Two" />
        <Tab label="Item Three" />
      </Tabs>
    </Box>
  );
}
```

Scrollable tabs

Automatic scroll buttons

Use the `variant="scrollable"` and `scrollButtons="auto"` props to display left and right scroll buttons on desktop that are hidden on mobile:

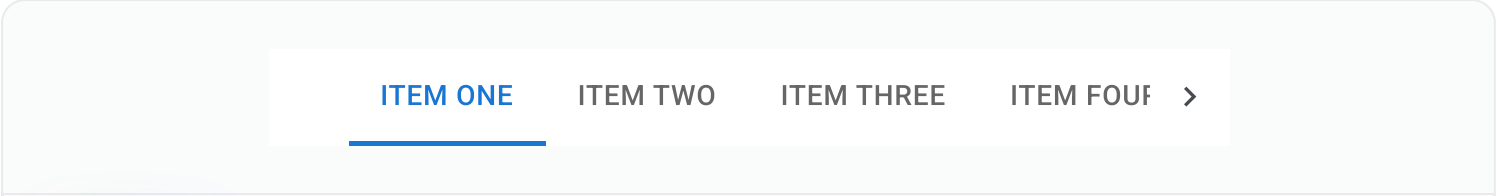


Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';

export default function ScrollableTabsButtonAuto() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ maxWidth: { xs: 320, sm: 480 }, bgcolor: 'background.paper' }}>
      <Tabs
        value={value}
        onChange={handleChange}
        variant="scrollable"
        scrollButtons="auto"
        aria-label="scrollable auto tabs example"
      >
        <Tab label="Item One" />
        <Tab label="Item Two" />
        <Tab label="Item Three" />
        <Tab label="Item Four" />
        <Tab label="Item Five" />
        <Tab label="Item Six" />
        <Tab label="Item Seven" />
      </Tabs>
    </Box>
  );
}
```

Forced scroll buttons

Apply `scrollButtons={true}` and the `allowScrollButtonsMobile` prop to display the left and right scroll buttons on all viewports:



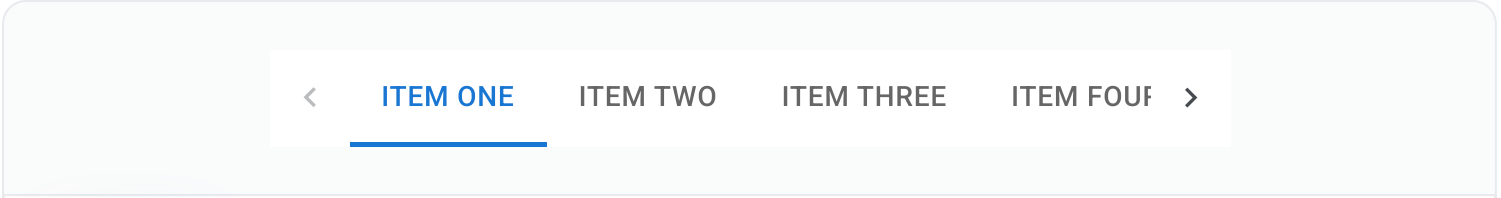
Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';

export default function ScrollableTabsButtonForce() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ maxWidth: { xs: 320, sm: 480 }, bgcolor: 'background.paper' }}>
      <Tabs
        value={value}
        onChange={handleChange}
        variant="scrollable"
        scrollButtons
        allowScrollButtonsMobile
        aria-label="scrollable force tabs example"
      >
        <Tab label="Item One" />
        <Tab label="Item Two" />
        <Tab label="Item Three" />
        <Tab label="Item Four" />
        <Tab label="Item Five" />
        <Tab label="Item Six" />
        <Tab label="Item Seven" />
      </Tabs>
    </Box>
  );
}
```

If you want to make sure the buttons are always visible, you should customize the opacity.

```css
.MuiTabs-scrollButtons.Mui-disabled {
  opacity: 0.3;
}
```

Copy

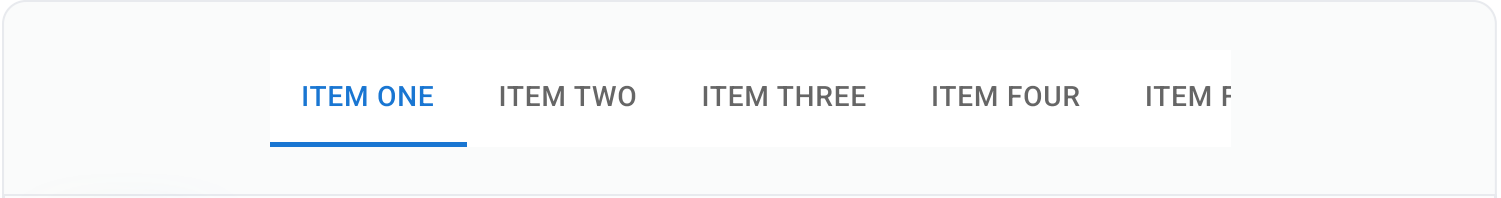


Edit in Chat | JS | TS | Hide code

```tsx
import * as React from 'react';
import Box from '@mui/material/Box';
import Tabs, { tabsClasses } from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';

export default function ScrollableTabsButtonVisible() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box
      sx={{
        flexGrow: 1,
        maxWidth: { xs: 320, sm: 480 },
        bgcolor: 'background.paper',
      }}
    >
      <Tabs
        value={value}
        onChange={handleChange}
        variant="scrollable"
        scrollButtons
        aria-label="visible arrows tabs example"
        sx={{
          [`& .${tabsClasses.scrollButtons}`]: {
            '&.Mui-disabled': { opacity: 0.3 },
          },
        }}
      >
        <Tab label="Item One" />
        <Tab label="Item Two" />
        <Tab label="Item Three" />
        <Tab label="Item Four" />
        <Tab label="Item Five" />
        <Tab label="Item Six" />
```

Prevent scroll buttons

Left and right scroll buttons are never be presented with `scrollButtons={false}`. All scrolling must be initiated through user agent scrolling mechanisms (for example left/right swipe, shift mouse wheel, etc.)

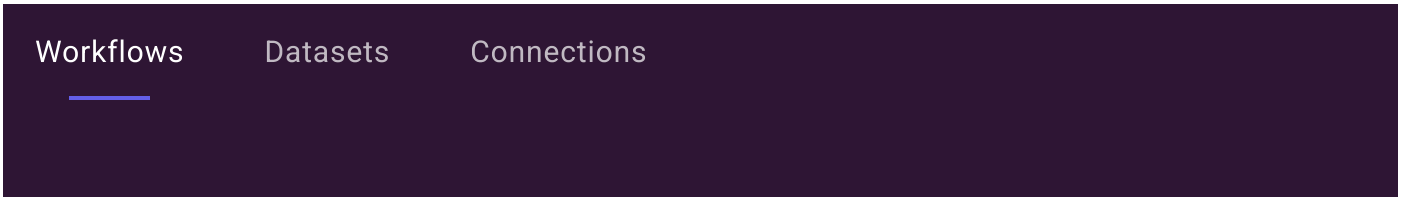


Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';

export default function ScrollableTabsButtonPrevent() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ maxWidth: { xs: 320, sm: 480 }, bgcolor: 'background.paper' }}>
      <Tabs
        value={value}
        onChange={handleChange}
        variant="scrollable"
        scrollButtons={false}
        aria-label="scrollable prevent tabs example"
      >
        <Tab label="Item One" />
        <Tab label="Item Two" />
        <Tab label="Item Three" />
        <Tab label="Item Four" />
        <Tab label="Item Five" />
        <Tab label="Item Six" />
        <Tab label="Item Seven" />
      </Tabs>
    </Box>
  );
}
```

Customization

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



Edit in Chat | JS | TS | Hide code

```tsx
import * as React from 'react';
import { styled } from '@mui/material/styles';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';

const AntTabs = styled(Tabs)({
  borderBottom: '1px solid #e8e8e8',
  '& .MuiTabs-indicator': {
    backgroundColor: '#1890ff',
  },
});

const AntTab = styled((props: StyledTabProps) => <Tab disableRipple {...props} />)(
  ({ theme }) => ({
    textTransform: 'none',
    minWidth: 0,
    [theme.breakpoints.up('sm')]: {
      minWidth: 0,
    },
    fontWeight: theme.typography.fontWeightRegular,
    marginRight: theme.spacing(1),
    color: 'rgba(0, 0, 0, 0.85)',
    fontFamily: [
      '-apple-system',
      'BlinkMacSystemFont',
      '"Segoe UI"',
      'Roboto',
      '"Helvetica Neue"',
      'Arial',
      'sans-serif',
      '"Apple Color Emoji"',
      '"Segoe UI Emoji"',
      '"Segoe UI Symbol"',
    ].join(','),
    '&:hover': {
      color: '#40a9ff',
      opacity: 1,
```

🎨 If you are looking for inspiration, you can check [MUI Treasury's customization examples](#).

Vertical tabs

To make vertical tabs instead of default horizontal ones, there is `orientation="vertical"`:

Edit in Chat | JS | TS | Hide code

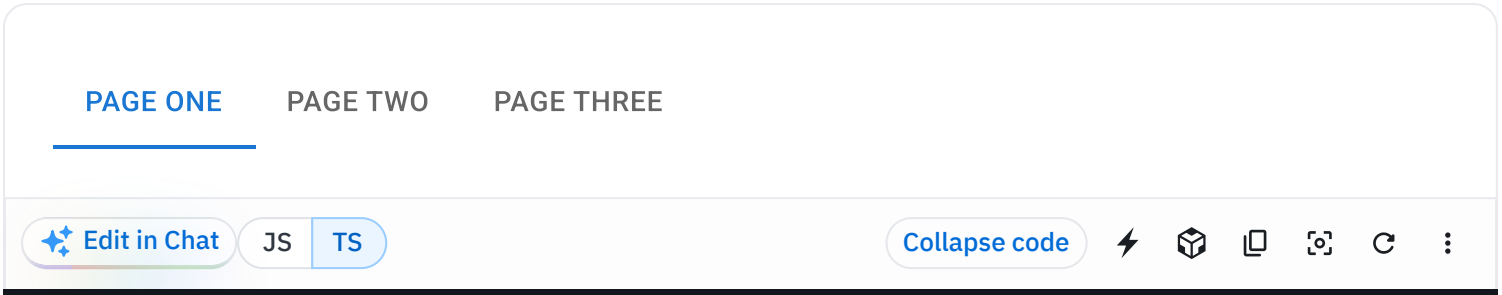
```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';

interface TabPanelProps {
  children?: React.ReactNode;
  index: number;
  value: number;
}

function TabPanel(props: TabPanelProps) {
  const { children, value, index, ...other } = props;

  return (
    <div
      role="tabpanel"
      hidden={value !== index}
      id={`vertical-tabpanel-${index}`}
      aria-labelledby={`vertical-tab-${index}`}
      {...other}
    >
      {value === index && (
        <Box sx={{ p: 3 }}>
          <Typography>{children}</Typography>
        </Box>
      )}
    </div>
  );
}

function a11yProps(index: number) {
  return {
    id: `vertical-tab-${index}`,
    'aria-controls': `vertical-tabpanel-${index}`,
```

```
};
```

Note that you can restore the scrollbar with `visibleScrollbar`.

Nav tabs

By default, tabs use a `button` element, but you can provide your custom tag or component. Here's an example of implementing tabbed navigation:



Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Box from '@mui/material/Box';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';

function samePageLinkNavigation(
  event: React.MouseEvent<HTMLAnchorElement, MouseEvent>,
) {
  if (
    event.defaultPrevented ||
    event.button !== 0 || // ignore everything but left-click
    event.metaKey ||
    event.ctrlKey ||
    event.altKey ||
    event.shiftKey
  ) {
    return false;
  }
  return true;
}

interface LinkTabProps {
  label?: string;
  href?: string;
  selected?: boolean;
}

function LinkTab(props: LinkTabProps) {
  return (
    <Tab
      component="a"
      onClick={(event: React.MouseEvent<HTMLAnchorElement, MouseEvent>) => {
        // Routing libraries handle this, you can remove the onClick handle when using them.
        if (samePageLinkNavigation(event)) {
          event.preventDefault();
```

```
        }
      }}
      aria-current={props.selected && 'page'}
```

Third-party routing library

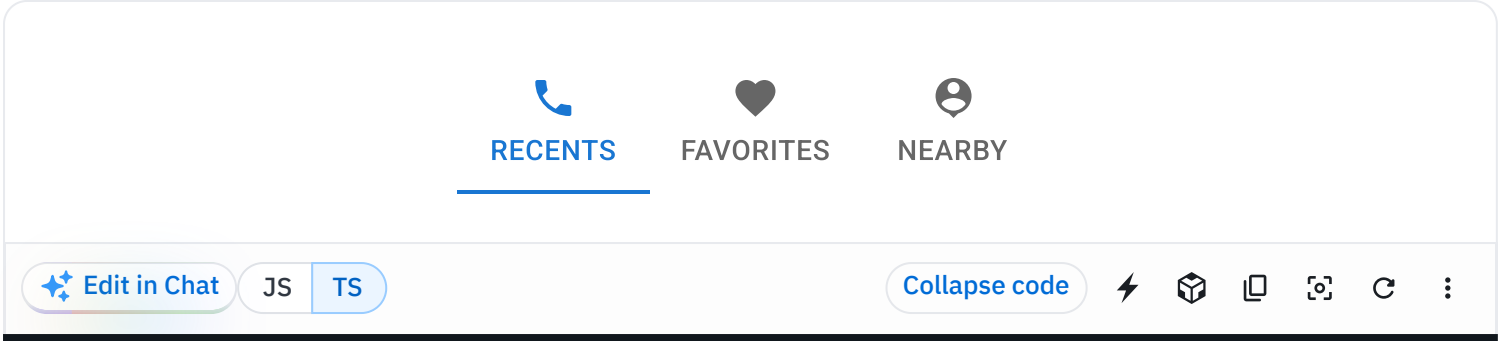
One frequent use case is to perform navigation on the client only, without an HTTP round-trip to the server. The `Tab` component provides the `component` prop to handle this use case. Here is a more detailed guide.

Icon tabs

Tab labels may be either all icons or all text.

Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import PhoneIcon from '@mui/icons-material/Phone';
import FavoriteIcon from '@mui/icons-material/Favorite';
import PersonPinIcon from '@mui/icons-material/PersonPin';

export default function IconTabs() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Tabs value={value} onChange={handleChange} aria-label="icon tabs example">
      <Tab icon={<PhoneIcon />} aria-label="phone" />
      <Tab icon={<FavoriteIcon />} aria-label="favorite" />
      <Tab icon={<PersonPinIcon />} aria-label="person" />
    </Tabs>
  );
}
```

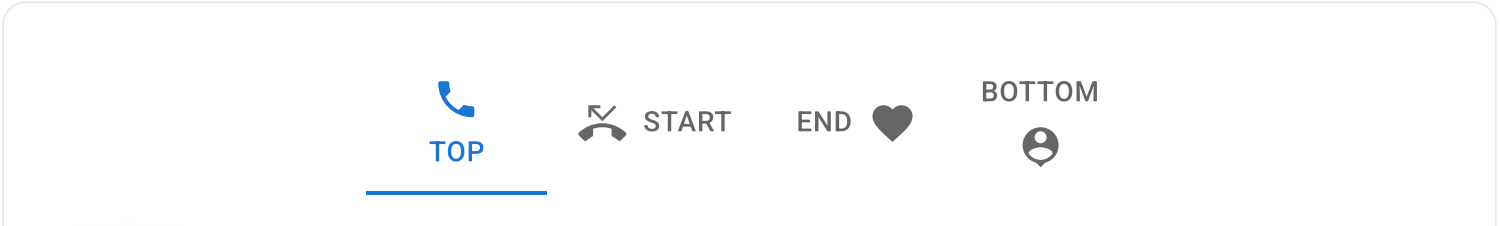


Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import PhoneIcon from '@mui/icons-material/Phone';
import FavoriteIcon from '@mui/icons-material/Favorite';
import PersonPinIcon from '@mui/icons-material/PersonPin';

export default function IconLabelTabs() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Tabs value={value} onChange={handleChange} aria-label="icon label tabs example">
      <Tab icon={<PhoneIcon />} label="RECENTS" />
      <Tab icon={<FavoriteIcon />} label="FAVORITES" />
      <Tab icon={<PersonPinIcon />} label="NEARBY" />
    </Tabs>
  );
}
```

Icon position

By default, the icon is positioned at the `top` of a tab. Other supported positions are `start`, `end`, `bottom`.



Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import PhoneIcon from '@mui/icons-material/Phone';
import FavoriteIcon from '@mui/icons-material/Favorite';
```

```
import PersonPinIcon from '@mui/icons-material/PersonPin';
import PhoneMissedIcon from '@mui/icons-material/PhoneMissed';

export default function IconPositionTabs() {
  const [value, setValue] = React.useState(0);

  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Tabs
      value={value}
      onChange={handleChange}
      aria-label="icon position tabs example"
    >
      <Tab icon={<PhoneIcon />} label="top" />
      <Tab icon={<PhoneMissedIcon />} iconPosition="start" label="start" />
      <Tab icon={<FavoriteIcon />} iconPosition="end" label="end" />
      <Tab icon={<PersonPinIcon />} iconPosition="bottom" label="bottom" />
    </Tabs>
  );
}
```

Accessibility

(WAI-ARIA: https://www.w3.org/WAI/ARIA/apg/patterns/tabs/)

The following steps are needed in order to provide necessary information for assistive technologies:

1. Label `Tabs` via `aria-label` or `aria-labelledby`.
2. `Tab`s need to be connected to their corresponding `[role="tabpanel"]` by setting the correct `id`, `aria-controls` and `aria-labelledby`.

An example for the current implementation can be found in the demos on this page. We've also published an experimental API in `@mui/lab` that does not require extra work.

Keyboard navigation

The components implement keyboard navigation using the "manual activation" behavior. If you want to switch to the "selection automatically follows focus" behavior you have to pass `selectionFollowsFocus` to the `Tabs` component. The WAI-ARIA authoring practices have a detailed guide on how to decide when to make selection automatically follow focus.

Demo

The following two demos only differ in their keyboard navigation behavior. Focus a tab and navigate with arrow keys to notice the difference, for example `Arrow Left`.

```
/* Tabs where selection follows focus */
<Tabs selectionFollowsFocus />
```

Copy

ITEM ONE ITEM TWO ITEM THREE

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Tabs from '@mui/material/Tabs';
import Tab from '@mui/material/Tab';
import Box from '@mui/material/Box';

export default function AccessibleTabs1() {
  const [value, setValue] = React.useState(0);
  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: '100%' }}>
      <Tabs
        onChange={handleChange}
        value={value}
        aria-label="Tabs where selection follows focus"
        selectionFollowsFocus
      >
        <Tab label="Item One" />
        <Tab label="Item Two" />
        <Tab label="Item Three" />
      </Tabs>
    </Box>
  );
}
```

```
/* Tabs where each tab needs to be selected manually */
<Tabs />
```

Copy

ITEM ONE ITEM TWO ITEM THREE

Edit in Chat JS TS Hide code

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

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

export default function AccessibleTabs2() {
  const [value, setValue] = React.useState(0);
  const handleChange = (event: React.SyntheticEvent, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: '100%' }}>
      <Tabs
        onChange={handleChange}
        value={value}
        aria-label="Tabs where each tab needs to be selected manually"
      >
        <Tab label="Item One" />
        <Tab label="Item Two" />
        <Tab label="Item Three" />
      </Tabs>
    </Box>
  );
}
```

API

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

- `<Tab />`
- `<TabContext />`
- `<TabList />`
- `<TabPanel />`
- `<TabScrollButton />`
- `<Tabs />`

‹ Stepper

Box ›

MUI • Blog ↗ • Store ↗