Stepper

Steppers convey progress through numbered steps. It provides a wizard-like workflow.

Steppers display progress through a sequence of logical and numbered steps. They may also be used for navigation. Steppers may display a transient feedback message after a step is saved.

- **Types of Steps**: Editable, Non-editable, Mobile, Optional
- **Types of Steppers**: Horizontal, Vertical, Linear, Non-linear

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

This component is no longer documented in the Material Design guidelines, but Material UI will continue to support it.

Introduction

The Stepper component displays progress through a sequence of logical and numbered steps. It supports horizontal and vertical orientation for desktop and mobile viewports.

Steppers are implemented using a collection of related components:

- Stepper: the container for the steps.
- Step: an individual step in the sequence.
- Step Label: a label for a Step.
- Step Content: optional content for a Step.
- Step Button: optional button for a Step.
- Step Icon: optional icon for a Step.
- Step Connector: optional customized connector between Steps.

Basics

```
import Stepper from '@mui/material/Stepper';
import Step from '@mui/material/Step';
import StepLabel from '@mui/material/StepLabel';
```

Copy

Horizontal stepper

Horizontal steppers are ideal when the contents of one step depend on an earlier step.

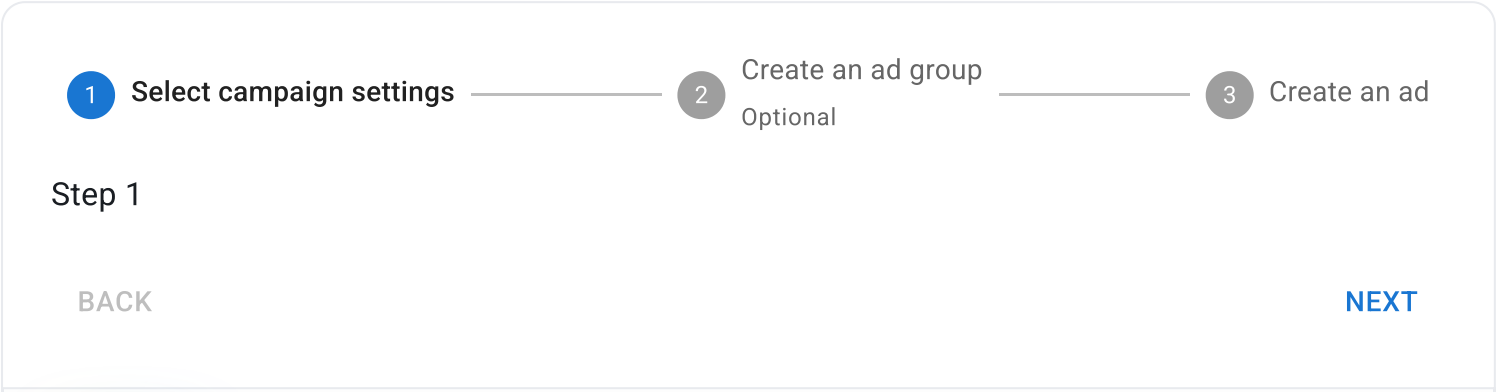
Avoid using long step names in horizontal steppers.

Linear

A linear stepper allows the user to complete the steps in sequence.

The `Stepper` can be controlled by passing the current step index (zero-based) as the `activeStep` prop. `Stepper` orientation is set using the `orientation` prop.

This example also shows the use of an optional step by placing the `optional` prop on the second `Step` component. Note that it's up to you to manage when an optional step is skipped. Once you've determined this for a particular step you must set `completed={false}` to signify that even though the active step index has gone beyond the optional step, it's not actually complete.



Edit in Chat | JS | TS | Hide code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Stepper from '@mui/material/Stepper';
import Step from '@mui/material/Step';
import StepLabel from '@mui/material/StepLabel';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';

const steps = ['Select campaign settings', 'Create an ad group', 'Create an ad'];

export default function HorizontalLinearStepper() {
  const [activeStep, setActiveStep] = React.useState(0);
  const [skipped, setSkipped] = React.useState(new Set<number>());

  const isStepOptional = (step: number) => {
    return step === 1;
  };
```

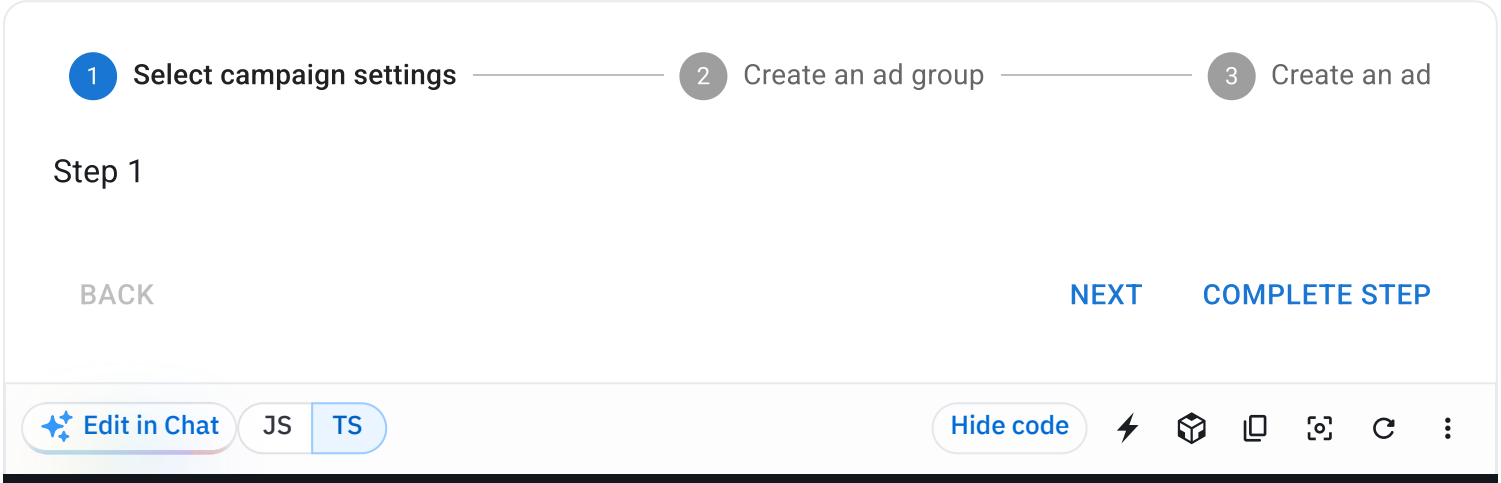
```tsx
  const isStepSkipped = (step: number) => {
    return skipped.has(step);
  };

  const handleNext = () => {
    let newSkipped = skipped;
    if (isStepSkipped(activeStep)) {
      newSkipped = new Set(newSkipped.values());
      newSkipped.delete(activeStep);
    }

    setActiveStep((prevActiveStep) => prevActiveStep + 1);
    setSkipped(newSkipped);
  };

  const handleBack = () => {
    setActiveStep((prevActiveStep) => prevActiveStep - 1);
  };

  const handleSkip = () => {
```

Non-linear

Non-linear steppers allow the user to enter a multi-step flow at any point.

This example is similar to the regular horizontal stepper, except steps are no longer automatically set to `disabled={true}` based on the `activeStep` prop.

The use of the `StepButton` here demonstrates clickable step labels, as well as setting the `completed` flag. However because steps can be accessed in a non-linear fashion, it's up to your own implementation to determine when all steps are completed (or even if they need to be completed).



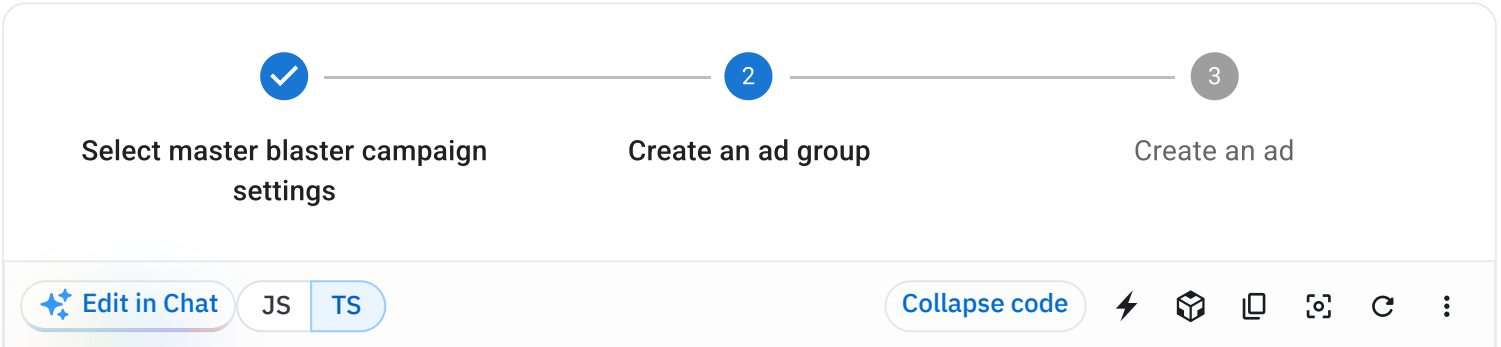
Edit in Chat | JS | TS | Hide code

```tsx
import * as React from 'react';
import Box from '@mui/material/Box';
import Stepper from '@mui/material/Stepper';
import Step from '@mui/material/Step';
import StepButton from '@mui/material/StepButton';
import Button from '@mui/material/Button';
import Typography from '@mui/material/Typography';
```

```tsx
const steps = ['Select campaign settings', 'Create an ad group', 'Create an ad'];

export default function HorizontalNonLinearStepper() {
  const [activeStep, setActiveStep] = React.useState(0);
  const [completed, setCompleted] = React.useState<{
    [k: number]: boolean;
  }>({});

  const totalSteps = () => {
    return steps.length;
  };

  const completedSteps = () => {
    return Object.keys(completed).length;
  };

  const isLastStep = () => {
    return activeStep === totalSteps() - 1;
  };

  const allStepsCompleted = () => {
    return completedSteps() === totalSteps();
  };

  const handleNext = () => {
    const newActiveStep =
      isLastStep() && !allStepsCompleted()
        ? // It's the last step, but not all steps have been completed,
          // find the first step that has been completed
          steps.findIndex((step, i) => !(i in completed))
```

Alternative label

Labels can be placed below the step icon by setting the `alternativeLabel` prop on the `Stepper` component.

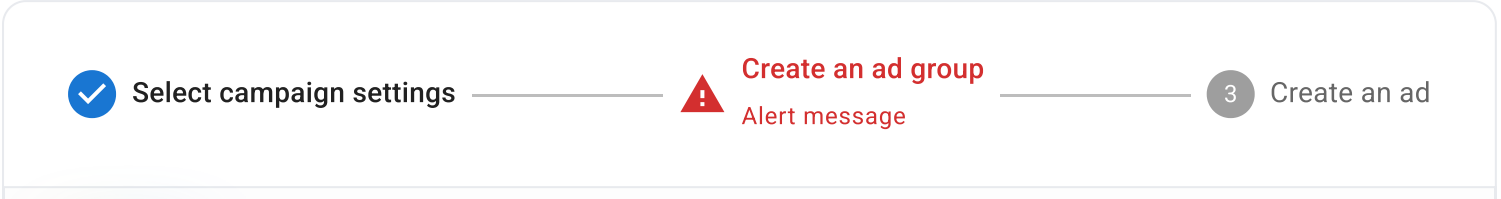


Edit in Chat JS TS Collapse code

```tsx
import Box from '@mui/material/Box';
import Stepper from '@mui/material/Stepper';
import Step from '@mui/material/Step';
import StepLabel from '@mui/material/StepLabel';

const steps = [
  'Select master blaster campaign settings',
  'Create an ad group',
```

```
  'Create an ad',
];

export default function HorizontalLinearAlternativeLabelStepper() {
  return (
    <Box sx={{ width: '100%' }}>
      <Stepper activeStep={1} alternativeLabel>
        {steps.map((label) => (
          <Step key={label}>
            <StepLabel>{label}</StepLabel>
          </Step>
        ))}
      </Stepper>
    </Box>
  );
}
```

Error step



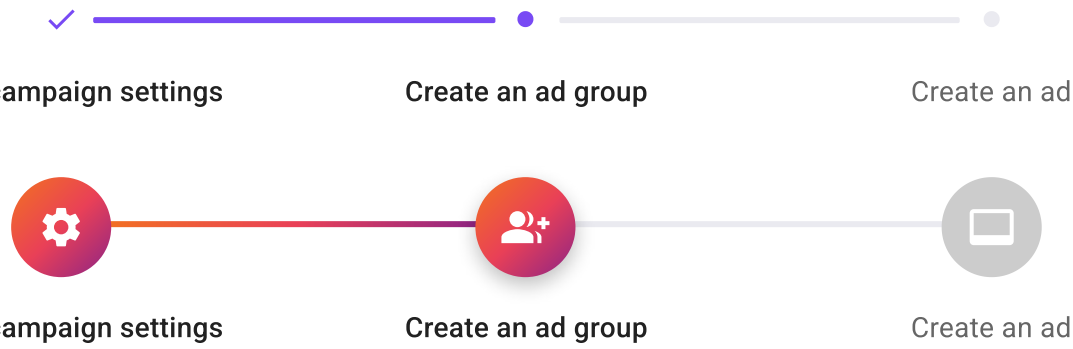
Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Stepper from '@mui/material/Stepper';
import Step from '@mui/material/Step';
import StepLabel from '@mui/material/StepLabel';
import Typography from '@mui/material/Typography';

const steps = ['Select campaign settings', 'Create an ad group', 'Create an ad'];

export default function HorizontalStepperWithError() {
  const isStepFailed = (step: number) => {
    return step === 1;
  };

  return (
    <Box sx={{ width: '100%' }}>
      <Stepper activeStep={1}>
        {steps.map((label, index) => {
          const labelProps: {
            optional?: React.ReactNode;
            error?: boolean;
          } = {};
          if (isStepFailed(index)) {
            labelProps.optional = (
              <Typography variant="caption" color="error">
                Alert message
              </Typography>
```

```
            );
            labelProps.error = true;
          }

          return (
            <Step key={label}>
              <StepLabel {...labelProps}>{label}</StepLabel>
            </Step>
          );
        })}
      </Stepper>
```

Customized horizontal stepper

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



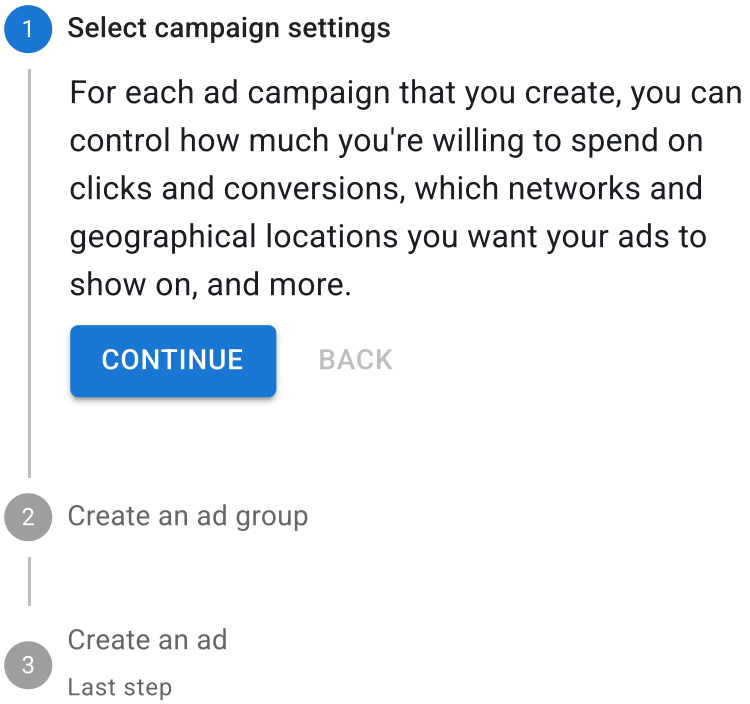
Edit in Chat | JS | TS | Collapse code

```
import * as React from 'react';
import { styled } from '@mui/material/styles';
import Stack from '@mui/material/Stack';
import Stepper from '@mui/material/Stepper';
import Step from '@mui/material/Step';
import StepLabel from '@mui/material/StepLabel';
import Check from '@mui/icons-material/Check';
import SettingsIcon from '@mui/icons-material/Settings';
import GroupAddIcon from '@mui/icons-material/GroupAdd';
import VideoLabelIcon from '@mui/icons-material/VideoLabel';
import StepConnector, { stepConnectorClasses } from '@mui/material/StepConnector';
import { StepIconProps } from '@mui/material/StepIcon';

const QontoConnector = styled(StepConnector)(({ theme }) => ({
  [`&.${stepConnectorClasses.alternativeLabel}`]: {
    top: 10,
    left: 'calc(-50% + 16px)',
    right: 'calc(50% + 16px)',
  },
  [`&.${stepConnectorClasses.active}`]: {
    [`& .${stepConnectorClasses.line}`]: {
      borderColor: '#784af4',
```

```
    },
  },
  [`&.${stepConnectorClasses.completed}`]: {
    [`& .${stepConnectorClasses.line}`]: {
      borderColor: '#784af4',
    },
  },
  [`& .${stepConnectorClasses.line}`]: {
    borderColor: '#eaeaf0',
    borderTopWidth: 3,
    borderRadius: 1,
    ...theme.applyStyles('dark', {
      borderColor: theme.palette.grey[800],
    }),
  },
}));
```

Vertical stepper

Vertical steppers are designed for narrow screen sizes. They are ideal for mobile. All the features of the horizontal stepper can be implemented.



Edit in Chat | JS | TS | Hide code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Stepper from '@mui/material/Stepper';
import Step from '@mui/material/Step';
import StepLabel from '@mui/material/StepLabel';
import StepContent from '@mui/material/StepContent';
```

```
import Button from '@mui/material/Button';
import Paper from '@mui/material/Paper';
import Typography from '@mui/material/Typography';

const steps = [
  {
    label: 'Select campaign settings',
    description: `For each ad campaign that you create, you can control how much
              you're willing to spend on clicks and conversions, which networks
              and geographical locations you want your ads to show on, and more.`,
  },
  {
    label: 'Create an ad group',
    description:
      'An ad group contains one or more ads which target a shared set of keywords.',
  },
  {
    label: 'Create an ad',
    description: `Try out different ad text to see what brings in the most customers,
              and learn how to enhance your ads using features like ad extensions.
              If you run into any problems with your ads, find out how to tell if
              they're running and how to resolve approval issues.`,
  },
];

export default function VerticalLinearStepper() {
  const [activeStep, setActiveStep] = React.useState(0);

  const handleNext = () => {
    setActiveStep((prevActiveStep) => prevActiveStep + 1);
  };
```

Performance

The content of a step is unmounted when closed. If you need to make the content available to search engines or render expensive component trees inside your modal while optimizing for interaction responsiveness it might be a good idea to keep the step mounted with:

```
<StepContent slotProps={{ transition: { unmountOnExit: false } }} />
```

Copy

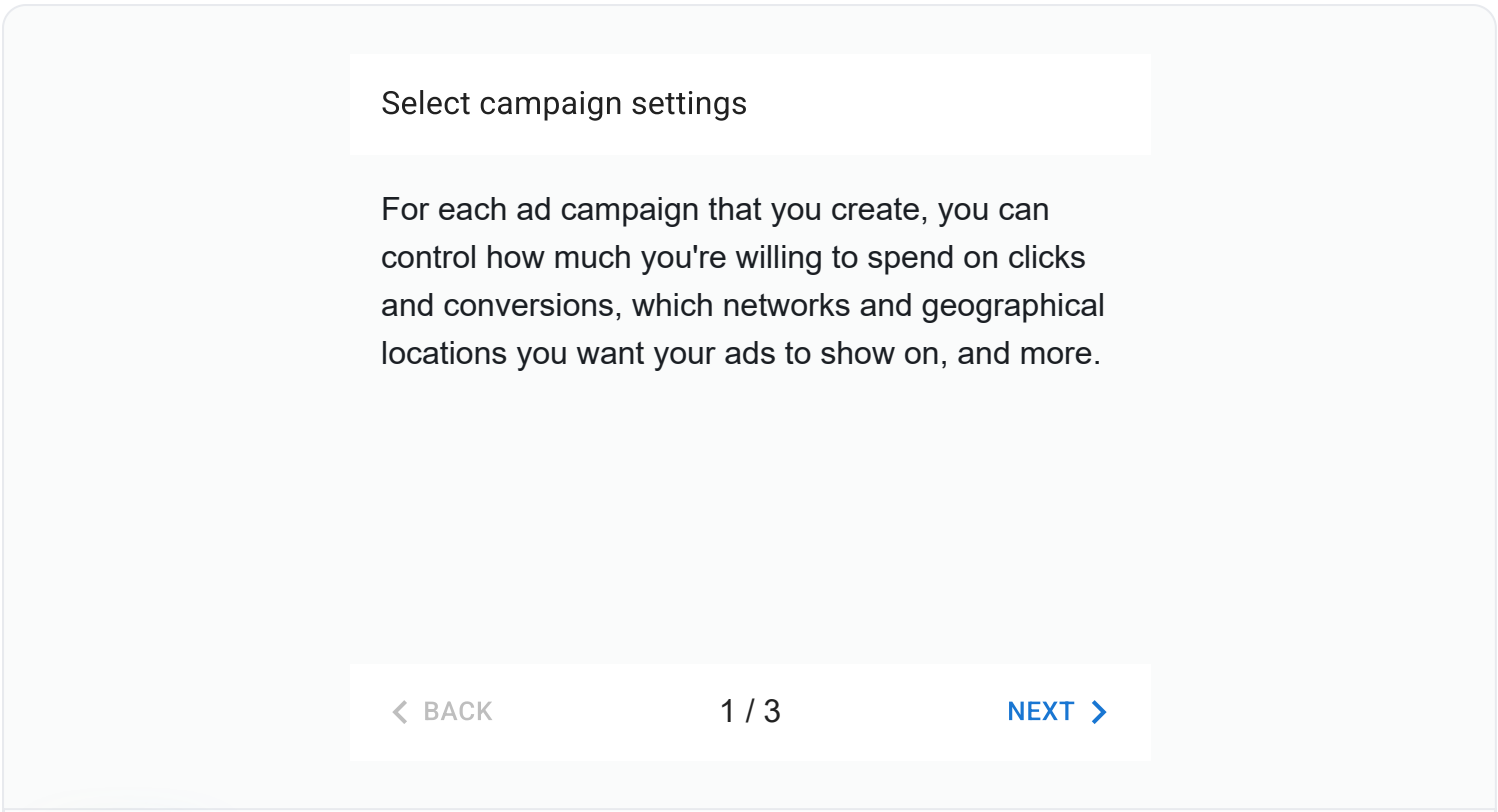
Mobile stepper

This component implements a compact stepper suitable for a mobile device. It has more limited functionality than the vertical stepper. See mobile steps for its inspiration.

The mobile stepper supports three variants to display progress through the available steps: text, dots, and progress.

Text

The current step and total number of steps are displayed as text.



Edit in Chat | JS | TS | Hide code

```tsx
import * as React from 'react';
import Box from '@mui/material/Box';
import { useTheme } from '@mui/material/styles';
import MobileStepper from '@mui/material/MobileStepper';
import Paper from '@mui/material/Paper';
import Typography from '@mui/material/Typography';
import Button from '@mui/material/Button';
import KeyboardArrowLeft from '@mui/icons-material/KeyboardArrowLeft';
import KeyboardArrowRight from '@mui/icons-material/KeyboardArrowRight';

const steps = [
  {
    label: 'Select campaign settings',
    description: `For each ad campaign that you create, you can control how much
              you're willing to spend on clicks and conversions, which networks
              and geographical locations you want your ads to show on, and more.`,
  },
  {
    label: 'Create an ad group',
    description:
      'An ad group contains one or more ads which target a shared set of keywords.',
  },
  {
    label: 'Create an ad',
    description: `Try out different ad text to see what brings in the most customers,
              and learn how to enhance your ads using features like ad extensions.
              If you run into any problems with your ads, find out how to tell if
              they're running and how to resolve approval issues.`,
  },
];
```

```
export default function TextMobileStepper() {
  const theme = useTheme();
  const [activeStep, setActiveStep] = React.useState(0);
  const maxSteps = steps.length;

  const handleNext = () => {
    setActiveStep((prevActiveStep) => prevActiveStep + 1);
```

Dots

Use dots when the number of steps is small.

Edit in Chat JS TS Hide code

```
import * as React from 'react';
import { useTheme } from '@mui/material/styles';
import MobileStepper from '@mui/material/MobileStepper';
import Button from '@mui/material/Button';
import KeyboardArrowLeft from '@mui/icons-material/KeyboardArrowLeft';
import KeyboardArrowRight from '@mui/icons-material/KeyboardArrowRight';

export default function DotsMobileStepper() {
  const theme = useTheme();
  const [activeStep, setActiveStep] = React.useState(0);

  const handleNext = () => {
    setActiveStep((prevActiveStep) => prevActiveStep + 1);
  };

  const handleBack = () => {
    setActiveStep((prevActiveStep) => prevActiveStep - 1);
  };

  return (
    <MobileStepper
      variant="dots"
      steps={6}
      position="static"
      activeStep={activeStep}
      sx={{ maxWidth: 400, flexGrow: 1 }}
      nextButton={
        <Button size="small" onClick={handleNext} disabled={activeStep === 5}>
          Next
          {theme.direction === 'rtl' ? (
            <KeyboardArrowLeft />
          ) : (
            <KeyboardArrowRight />
          )}
```

```
      </Button>
    }
    backButton={
      <Button size="small" onClick={handleBack} disabled={activeStep === 0}>
```

Progress

Use a progress bar when there are many steps, or if there are steps that need to be inserted during the process (based on responses to earlier steps).

Edit in Chat JS TS Hide code

```tsx
import * as React from 'react';
import { useTheme } from '@mui/material/styles';
import MobileStepper from '@mui/material/MobileStepper';
import Button from '@mui/material/Button';
import KeyboardArrowLeft from '@mui/icons-material/KeyboardArrowLeft';
import KeyboardArrowRight from '@mui/icons-material/KeyboardArrowRight';

export default function ProgressMobileStepper() {
  const theme = useTheme();
  const [activeStep, setActiveStep] = React.useState(0);

  const handleNext = () => {
    setActiveStep((prevActiveStep) => prevActiveStep + 1);
  };

  const handleBack = () => {
    setActiveStep((prevActiveStep) => prevActiveStep - 1);
  };

  return (
    <MobileStepper
      variant="progress"
      steps={6}
      position="static"
      activeStep={activeStep}
      sx={{ maxWidth: 400, flexGrow: 1 }}
      nextButton={
        <Button size="small" onClick={handleNext} disabled={activeStep === 5}>
          Next
          {theme.direction === 'rtl' ? (
            <KeyboardArrowLeft />
          ) : (
            <KeyboardArrowRight />
          )}
        </Button>
      }
```

```
      backButton={
        <Button size="small" onClick={handleBack} disabled={activeStep === 0}>
```

API

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

- `<MobileStepper />`
- `<Step />`
- `<StepButton />`
- `<StepConnector />`
- `<StepContent />`
- `<StepIcon />`
- `<StepLabel />`
- `<Stepper />`

Edit this page

Was this page helpful?

Speed Dial

Tabs