

Slider

Sliders allow users to make selections from a range of values.



Check out the latest remote job listings from Authentic Jobs.

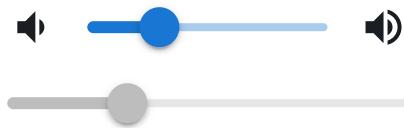
ads via Carbon

Sliders reflect a range of values along a bar, from which users may select a single value. They are ideal for adjusting settings such as volume, brightness, or applying image filters.

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

Continuous sliders

Continuous sliders allow users to select a value along a subjective range.

[Edit in Chat](#)

JS

TS

[Collapse code](#)

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Stack from '@mui/material/Stack';
import Slider from '@mui/material/Slider';
import VolumeDown from '@mui/icons-material/VolumeDown';
import VolumeUp from '@mui/icons-material/VolumeUp';

export default function ContinuousSlider() {
  const [value, setValue] = React.useState<number>(30);

  const handleChange = (event: Event, newValue: number) => {
    setValue(newValue);
  };

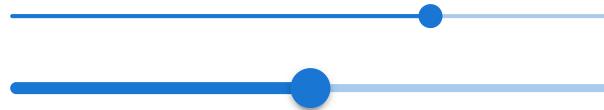
  return (
    <Box sx={{ width: 200 }}>
      <Stack spacing={2} direction="row" sx={{ alignItems: 'center', mb: 1 }}>
```

```
<VolumeDown />
<Slider aria-label="Volume" value={value} onChange={handleChange} />
<VolumeUp />
</Stack>
<Slider disabled defaultValue={30} aria-label="Disabled slider" />
</Box>
);
}
```

Sizes

+

For smaller slider, use the prop `size="small"`.



Edit in Chat

JS

TS

Collapse code



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

export default function SliderSizes() {
  return (
    <Box sx={{ width: 300 }}>
      <Slider
        size="small"
        defaultValue={70}
        aria-label="Small"
        valueLabelDisplay="auto"
      />
      <Slider defaultValue={50} aria-label="Default" valueLabelDisplay="auto" />
    </Box>
  );
}
```

Discrete sliders

+

Discrete sliders can be adjusted to a specific value by referencing its value indicator. You can generate a mark for each step with `marks={true}`.

[Edit in Chat](#)

JS

TS

[Collapse code](#)

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

function valuetext(value: number) {
  return `${value}°C`;
}

export default function DiscreteSlider() {
  return (
    <Box sx={{ width: 300 }}>
      <Slider
        aria-label="Temperature"
        defaultValue={30}
        getAriaValueText={valuetext}
        valueLabelDisplay="auto"
        shiftStep={30}
        step={10}
        marks
        min={10}
        max={110}
      />
      <Slider defaultValue={30} step={10} marks min={10} max={110} disabled />
    </Box>
  );
}
```

Small steps



You can change the default step increment. Make sure to adjust the `shiftStep` prop (the granularity with which the slider can step when using Page Up/Down or Shift + Arrow Up/Down) to a value divisible by the `step`.

[Edit in Chat](#)

JS

TS

[Collapse code](#)

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

function valuetext(value: number) {
```

```
return `${value}°C`;  
}  
  
export default function DiscreteSliderSteps() {  
  return (  
    <Box sx={{ width: 300 }}>  
      <Slider  
        aria-label="Small steps"  
        defaultValue={0.0000005}  
        getAriaValueText={valuetext}  
        step={0.0000001}  
        marks  
        min={-0.0000005}  
        max={0.0000001}  
        valueLabelDisplay="auto"  
      />  
    </Box>  
  );  
}
```

Custom marks



You can have custom marks by providing a rich array to the `marks` prop.



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

[Collapse code](#)

```
import Box from '@mui/material/Box';  
import Slider from '@mui/material/Slider';  
  
const marks = [  
  {  
    value: 0,  
    label: '0°C',  
  },  
  {  
    value: 20,  
    label: '20°C',  
  },  
  {  
    value: 37,  
    label: '37°C',  
  },  
  {  
    value: 100,  
    label: '100°C',  
  },  
];
```

```
function valuetext(value: number) {
  return `${value}°C`;
}

export default function DiscreteSliderMarks() {
  return (
    <Box sx={{ width: 300 }}>
      <Slider
        aria-label="Custom marks"
        defaultValue={20}
        getAriaValueText={valuetext}
        step={10}
        valueLabelDisplay="auto"
        marks={marks}
      />
    </Box>
  );
}
```

Restricted values

You can restrict the selectable values to those provided with the `marks` prop with `step={null}`.



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

[Collapse code](#)

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

const marks = [
  {
    value: 0,
    label: '0°C',
  },
  {
    value: 20,
    label: '20°C',
  },
  {
    value: 37,
    label: '37°C',
  },
  {
    value: 100,
    label: '100°C',
  },
];

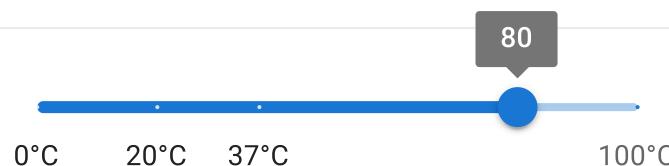
function valuetext(value: number) {
  return `${value}°C`;
}
```

```
}

export default function DiscreteSliderValues() {
  return (
    <Box sx={{ width: 300 }}>
      <Slider
        aria-label="Restricted values"
        defaultValue={20}
        getAriaValueText={valuetext}
        step={null}
        valueLabelDisplay="auto"
        marks={marks}
      />
    </Box>
  )
}
```

Label always visible

You can force the thumb label to be always visible with `valueLabelDisplay="on"`.



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

[Collapse code](#)

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

const marks = [
  {
    value: 0,
    label: '0°C',
  },
  {
    value: 20,
    label: '20°C',
  },
  {
    value: 37,
    label: '37°C',
  },
  {
    value: 100,
    label: '100°C',
  },
];

function valuetext(value: number) {
  return `${value}°C`;
}

export default function DiscreteSliderLabel() {
```

```
return (
  <Box sx={{ width: 300 }}>
    <Slider
      aria-label="Always visible"
      defaultValue={80}
      getAriaValueText={valuetext}
      step={10}
      marks={marks}
      valueLabelDisplay="on"
    />
  </Box>
```

Range slider

The slider can be used to set the start and end of a range by supplying an array of values to the `value` prop.



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

[Collapse code](#)

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

function valuetext(value: number) {
  return `${value}°C`;
}

export default function RangeSlider() {
  const [value, setValue] = React.useState<number[]>([20, 37]);

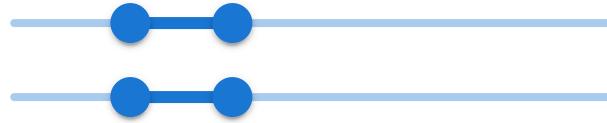
  const handleChange = (event: Event, newValue: number[]) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: 300 }}>
      <Slider
        getAriaLabel={() => 'Temperature range'}
        value={value}
        onChange={handleChange}
        valueLabelDisplay="auto"
        getAriaValueText={valuetext}
      />
    </Box>
  );
}
```

Minimum distance

+

You can enforce a minimum distance between values in the `onChange` event handler. By default, when you move the pointer over a thumb while dragging another thumb, the active thumb will swap to the hovered thumb. You can disable this behavior with the `disableSwap` prop. If you want the range to shift when reaching minimum distance, you can utilize the `activeThumb` parameter in `onChange`.

[Edit in Chat](#)

JS

TS

[Collapse code](#)

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

function valuetext(value: number) {
  return `${value}°C`;
}

const minDistance = 10;

export default function MinimumDistanceSlider() {
  const [value1, setValue1] = React.useState<number>([20, 37]);

  const handleChange1 = (event: Event, newValue: number[], activeThumb: number) => {
    if (activeThumb === 0) {
      setValue1([Math.min(newValue[0], value1[1] - minDistance), value1[1]]);
    } else {
      setValue1([value1[0], Math.max(newValue[1], value1[0] + minDistance)]);
    }
  };

  const [value2, setValue2] = React.useState<number>([20, 37]);

  const handleChange2 = (event: Event, newValue: number[], activeThumb: number) => {
    if (newValue[1] - newValue[0] < minDistance) {
      if (activeThumb === 0) {
        const clamped = Math.min(newValue[0], 100 - minDistance);
        setValue2([clamped, clamped + minDistance]);
      } else {
        const clamped = Math.max(newValue[1], minDistance);
        setValue2([clamped - minDistance, clamped]);
      }
    } else {
      setValue2(newValue);
    }
  };
}
```

Slider with input field

In this example, an input allows a discrete value to be set.



```
import * as React from 'react';
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Grid from '@mui/material/Grid';
import Typography from '@mui/material/Typography';
import Slider from '@mui/material/Slider';
import MuiInput from '@mui/material/Input';
import VolumeUp from '@mui/icons-material/VolumeUp';

const Input = styled(MuiInput)`  

  width: 42px;  

`;  
  

export default function InputSlider() {
  const [value, setValue] = React.useState(30);  
  

  const handleSliderChange = (event: Event, newValue: number) => {
    setValue(newValue);
  };  
  

  const handleInputChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setValue(event.target.value === '' ? 0 : Number(event.target.value));
  };  
  

  const handleBlur = () => {
    if (value < 0) {
      setValue(0);
    } else if (value > 100) {
      setValue(100);
    }
  };
  

  return (
    <Box sx={{ width: 250 }}>
      <Typography id="input-slider" gutterBottom>
        Volume
      </Typography>
      <Slider
        value={value}
        onChange={handleSliderChange}
        valueLabel={value}
      >
        <VolumeUp />
      </Slider>
      <Input value={value} onChange={handleInputChange} />
    </Box>
  );
}
```

Color

+



Edit in Chat JS TS

Collapse code

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

function valuetext(value: number) {
  return `${value}°C`;
}

export default function ColorSlider() {
  return (
    <Box sx={{ width: 300 }}>
      <Slider
        aria-label="Temperature"
        defaultValue={30}
        getAriaValueText={valuetext}
        color="secondary"
      />
    </Box>
  );
}
```

Customization

+

Here are some examples of customizing the component. You can learn more about this in the [overrides documentation page](#).

iOS

60



prettos.fr



Tooltip value label

Airbnb

[Edit in Chat](#)

JS

TS

[Hide code](#)

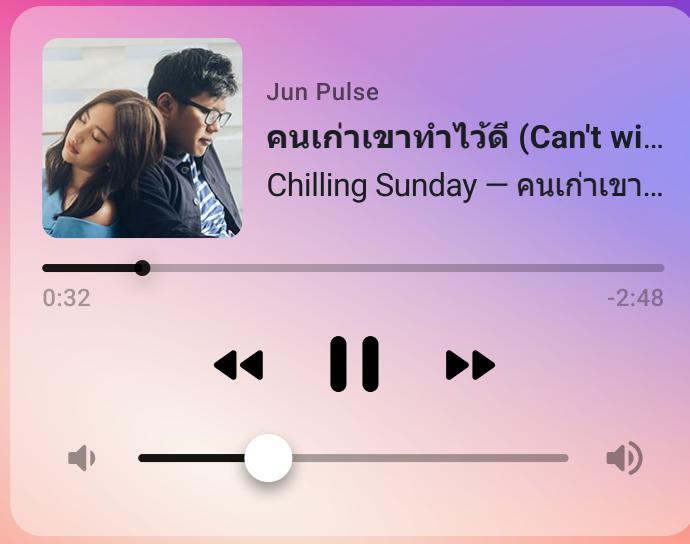
```
import * as React from 'react';
import Slider, { SliderThumb, SliderValueLabelProps } from '@mui/material/Slider';
import { styled } from '@mui/material/styles';
import Typography from '@mui/material/Typography';
import Tooltip from '@mui/material/Tooltip';
import Box from '@mui/material/Box';

function ValueLabelComponent(props: SliderValueLabelProps) {
  const { children, value } = props;

  return (
    <Tooltip enterTouchDelay={0} placement="top" title={value}>
      {children}
    </Tooltip>
  );
}

const iOSBoxShadow =
  '0 3px 1px rgba(0,0,0,0.1),0 4px 8px rgba(0,0,0,0.13),0 0 0 1px rgba(0,0,0,0.02)';

const IOSSlider = styled(Slider)(({ theme }) => ({
  color: '#007bff',
  height: 5,
  padding: '15px 0',
  '& .MuiSlider-thumb': {
    height: 20,
    width: 20,
    backgroundColor: '#fff',
    boxShadow: '0 0 2px 0px rgba(0, 0, 0, 0.1)',
    '&:focus, &:hover, &.Mui-active': {
      boxShadow: '0px 0px 3px 1px rgba(0, 0, 0, 0.1)',
      // Reset on touch devices, it doesn't add specificity
      '@media (hover: none)': {
        boxShadow: iOSBoxShadow,
      },
    },
    '&:before': {
      boxShadow:
```

[Edit in Chat](#)

JS

TS

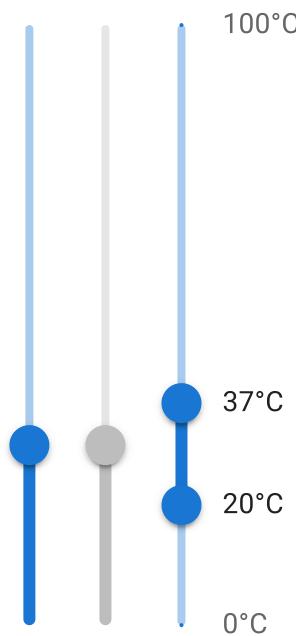
[Hide code](#)

```
import * as React from 'react';
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Typography from '@mui/material/Typography';
import Slider from '@mui/material/Slider';
import IconButton from '@mui/material/IconButton';
import Stack from '@mui/material/Stack';
import PauseRounded from '@mui/icons-material/PauseRounded';
import PlayArrowRounded from '@mui/icons-material/PlayArrowRounded';
import FastForwardRounded from '@mui/icons-material/FastForwardRounded';
import FastRewindRounded from '@mui/icons-material/FastRewindRounded';
import VolumeUpRounded from '@mui/icons-material/VolumeUpRounded';
import VolumeDownRounded from '@mui/icons-material/VolumeDownRounded';

const WallPaper = styled('div')({
  position: 'absolute',
  width: '100%',
  height: '100%',
  top: 0,
  left: 0,
  overflow: 'hidden',
  background: 'linear-gradient(rgb(255, 38, 142) 0%, rgb(255, 105, 79) 100%)',
  transition: 'all 500ms cubic-bezier(0.175, 0.885, 0.32, 1.275) 0s',
  '&::before': {
    content: '',
    width: '140%',
    height: '140%',
    position: 'absolute',
    top: '-40%',
    right: '-50%',
    background:
      'radial-gradient(at center center, rgb(62, 79, 249) 0%, rgba(62, 79, 249, 0) 64%)',
  },
  '&::after': {
    content: '',
    width: '140%',
    height: '140%',
```

Vertical sliders

Set the `orientation` prop to `"vertical"` to create vertical sliders. The thumb will track vertical movement instead of horizontal movement.



[Edit in Chat](#)

[JS](#) [TS](#)

[Hide code](#)



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

export default function VerticalSlider() {
  return (
    <Stack sx={{ height: 300 }} spacing={1} direction="row">
      <Slider
        aria-label="Temperature"
        orientation="vertical"
        getAriaValueText={getAriaValueText}
        valueLabelDisplay="auto"
        defaultValue={30}
      />
      <Slider
        aria-label="Temperature"
        orientation="vertical"
        defaultValue={30}
        valueLabelDisplay="auto"
        disabled
      />
      <Slider
        getAriaLabel={() => 'Temperature'}
        orientation="vertical"
        getAriaValueText={getAriaValueText}
        defaultValue={[20, 37]}
      />
    </Stack>
  );
}

function getAriaValueText() {
  const value = [20, 37];
  if (value[0] === value[1]) {
    return `${value[0]}°C`;
  }
  return `${value[0]}°C - ${value[1]}°C`;
}
```

```
    valueLabelDisplay="auto"
    marks={marks}
  />
</Stack>
);
}

function getAriaValueText(value: number) {
  return `${value}°C`;
}

const marks = [
  {
    value: 0,
    label: '0 min',
    thumb: true
  },
  {
    value: 100,
    label: '100 max',
    thumb: true
  }
]
```

⚠ Chrome versions below 124 implement `aria-orientation` incorrectly for vertical sliders and expose them as `'horizontal'` in the accessibility tree. ([Chromium issue #40736841](#))

The `-webkit-appearance: slider-vertical` CSS property can be used to correct this for these older versions, with the trade-off of causing a console warning in newer Chrome versions:

```
.MuiSlider-thumb input {
  -webkit-appearance: slider-vertical;
}
```

Copy

Marks placement

You can customize your slider by adding and repositioning marks for minimum and maximum values.



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

[Hide code](#)

⚡ ⚙️ 📁 🔍 ⌂ ⌂ ⌂ ⌂ ⌂

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Slider from '@mui/material/Slider';
import Typography from '@mui/material/Typography';

const MAX = 100;
const MIN = 0;
const marks = [
  {
    value: MIN,
    label: '',
    thumb: true
  },
  {
    value: 100,
    label: '100 max',
    thumb: true
  }
]
```

```

    value: MAX,
    label: '',
},
];

export default function CustomMarks() {
  const [val, setVal] = React.useState<number>(MIN);
  const handleChange = (_: Event, newValue: number) => {
    setVal(newValue);
};

return (
  <Box sx={{ width: 250 }}>
    <Slider
      marks={marks}
      step={10}
      value={val}
      valueLabelDisplay="auto"
      min={MIN}
      max={MAX}
      onChange={handleChange}
    />
    <Box sx={{ display: 'flex', justifyContent: 'space-between' }}>
      <Typography
        variant="body2"

```

Track



The track shows the range available for user selection.

Removed track



The track can be turned off with `track={false}`.

Removed track



Removed track range slider



[Edit in Chat](#)

[JS](#) [TS](#)

[Hide code](#)



```

import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Typography from '@mui/material/Typography';

```

```

import Slider from '@mui/material/Slider';

const Separator = styled('div')(
  ({ theme }) => `
    height: ${theme.spacing(3)};
  `,
);

const marks = [
  {
    value: 0,
    label: '0°C',
  },
  {
    value: 20,
    label: '20°C',
  },
  {
    value: 37,
    label: '37°C',
  },
  {
    value: 100,
    label: '100°C',
  },
];

```

```

function valuetext(value: number) {
  return `${value}°C`;
}

```

```

export default function TrackFalseSlider() {
  return (
    <Box sx={{ width: 250 }}>

```

Inverted track

The track can be inverted with `track="inverted"`.

Inverted track



Inverted track range



```

import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Typography from '@mui/material/Typography';
import Slider from '@mui/material/Slider';

const Separator = styled('div')(
  ({ theme }) =>
  height: ${theme.spacing(3)};
` ,
);

const marks = [
{
  value: 0,
  label: '0°C',
},
{
  value: 20,
  label: '20°C',
},
{
  value: 37,
  label: '37°C',
},
{
  value: 100,
  label: '100°C',
},
];
]

function valuetext(value: number) {
  return `${value}°C`;
}

export default function TrackInvertedSlider() {
  return (
    <Box sx={{ width: 250 }}>
      <Typography id="track-inverted-slider" gutterBottom>

```

Non-linear scale



You can use the `scale` prop to represent the `value` on a different scale.

In the following demo, the value x represents the value 2^x . Increasing x by one increases the represented value by factor 2.

Storage: 1 MB



```
import * as React from 'react';
import Box from '@mui/material/Box';
import Typography from '@mui/material/Typography';
import Slider from '@mui/material/Slider';

function valueLabelFormat(value: number) {
  const units = ['KB', 'MB', 'GB', 'TB'];

  let unitIndex = 0;
  let scaledValue = value;

  while (scaledValue >= 1024 && unitIndex < units.length - 1) {
    unitIndex += 1;
    scaledValue /= 1024;
  }

  return `${scaledValue} ${units[unitIndex]}`;
}

function calculateValue(value: number) {
  return 2 ** value;
}

export default function NonLinearSlider() {
  const [value, setValue] = React.useState<number>(10);

  const handleChange = (event: Event, newValue: number) => {
    setValue(newValue);
  };

  return (
    <Box sx={{ width: 250 }}>
      <Typography id="non-linear-slider" gutterBottom>
        Storage: {valueLabelFormat(calculateValue(value))}
      </Typography>
      <Slider
        value={value}
        min={5}
      >
    
```

Accessibility

(WAI-ARIA: <https://www.w3.org/WAI/ARIA/apg/patterns/slider-multithumb/>)

The component handles most of the work necessary to make it accessible. However, you need to make sure that:

- Each thumb has a user-friendly label (`aria-label`, `aria-labelledby` or `getAriaLabel` prop).

- Each thumb has a user-friendly text for its current value. This is not required if the value matches the semantics of the label. You can change the name with the `getAriaValueText` or `aria-valuetext` prop.

API



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

- [`<Slider />`](#)

Edit this page

Was this page helpful?

[Select](#)

[Switch](#) >
