Rating

Ratings provide insight regarding others' opinions and experiences, and can allow the user to submit a rating of their own.

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

Basic rating

Controlled

Uncontrolled

Read only

Disabled

No rating given

Edit in Chat | JS | TS | Hide code

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

export default function BasicRating() {
  const [value, setValue] = React.useState<number | null>(2);
```

```
  return (
    <Box sx={{ '& > legend': { mt: 2 } }}>
      <Typography component="legend">Controlled</Typography>
      <Rating
        name="simple-controlled"
        value={value}
        onChange={(event, newValue) => {
          setValue(newValue);
        }}
      />
      <Typography component="legend">Uncontrolled</Typography>
      <Rating
        name="simple-uncontrolled"
        onChange={(event, newValue) => {
          console.log(newValue);
        }}
        defaultValue={2}
      />
      <Typography component="legend">Read only</Typography>
      <Rating name="read-only" value={value} readOnly />
      <Typography component="legend">Disabled</Typography>
      <Rating name="disabled" value={value} disabled />
      <Typography component="legend">No rating given</Typography>
      <Rating name="no-value" value={null} />
    </Box>
  );
}
```

Rating precision

The rating can display any float number with the `value` prop. Use the `precision` prop to define the minimum increment value change allowed.

Edit in Chat | JS | TS | Collapse code

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

export default function HalfRating() {
  return (
    <Stack spacing={1}>
      <Rating name="half-rating" defaultValue={2.5} precision={0.5} />
      <Rating name="half-rating-read" defaultValue={2.5} precision={0.5} readOnly />
    </Stack>
```

```
  );
}
```

Hover feedback

You can display a label on hover to help the user pick the correct rating value. The demo uses the `onChangeActive` prop.

Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Rating from '@mui/material/Rating';
import Box from '@mui/material/Box';
import StarIcon from '@mui/icons-material/Star';

const labels: { [index: string]: string } = {
  0.5: 'Useless',
  1: 'Useless+',
  1.5: 'Poor',
  2: 'Poor+',
  2.5: 'Ok',
  3: 'Ok+',
  3.5: 'Good',
  4: 'Good+',
  4.5: 'Excellent',
  5: 'Excellent+',
};

function getLabelText(value: number) {
  return `${value} Star${value !== 1 ? 's' : ''}, ${labels[value]}`;
}

export default function HoverRating() {
  const [value, setValue] = React.useState<number | null>(2);
  const [hover, setHover] = React.useState(-1);

  return (
    <Box sx={{ width: 200, display: 'flex', alignItems: 'center' }}>
      <Rating
        name="hover-feedback"
        value={value}
        precision={0.5}
        getLabelText={getLabelText}
        onChange={(event, newValue) => {
          setValue(newValue);
        }}
        onChangeActive={(event, newHover) => {
```

```
        setHover(newHover);
```

Sizes

For larger or smaller ratings use the `size` prop.

Edit in Chat | JS | TS | Collapse code

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

export default function RatingSize() {
  return (
    <Stack spacing={1}>
      <Rating name="size-small" defaultValue={2} size="small" />
      <Rating name="size-medium" defaultValue={2} />
      <Rating name="size-large" defaultValue={2} size="large" />
    </Stack>
  );
}
```

Customization

Here are some examples of customizing the component. You can learn more about this in the overrides documentation page.

Custom icon and color

10 stars

Edit in Chat | JS | TS | Collapse code

```tsx
import { styled } from '@mui/material/styles';
import Box from '@mui/material/Box';
import Rating from '@mui/material/Rating';
import FavoriteIcon from '@mui/icons-material/Favorite';
import FavoriteBorderIcon from '@mui/icons-material/FavoriteBorder';
import Typography from '@mui/material/Typography';

const StyledRating = styled(Rating)({
  '& .MuiRating-iconFilled': {
    color: '#ff6d75',
  },
  '& .MuiRating-iconHover': {
    color: '#ff3d47',
  },
});

export default function CustomizedRating() {
  return (
    <Box sx={{ '& > legend': { mt: 2 } }}>
      <Typography component="legend">Custom icon and color</Typography>
      <StyledRating
        name="customized-color"
        defaultValue={2}
        getLabelText={(value: number) => `${value} Heart${value !== 1 ? 's' : ''}`}
        precision={0.5}
        icon={<FavoriteIcon fontSize="inherit" />}
        emptyIcon={<FavoriteBorderIcon fontSize="inherit" />}
      />
      <Typography component="legend">10 stars</Typography>
      <Rating name="customized-10" defaultValue={2} max={10} />
    </Box>
  );
}
```

Radio group

The rating is implemented with a radio group, set `highlightSelectedOnly` to restore the natural behavior.

Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import { styled } from '@mui/material/styles';
import Rating, { IconContainerProps } from '@mui/material/Rating';
import SentimentVeryDissatisfiedIcon from '@mui/icons-material/SentimentVeryDissatisfied';
import SentimentDissatisfiedIcon from '@mui/icons-material/SentimentDissatisfied';
import SentimentSatisfiedIcon from '@mui/icons-material/SentimentSatisfied';
```

```tsx
import SentimentSatisfiedAltIcon from '@mui/icons-material/SentimentSatisfiedAltOutlined';
import SentimentVerySatisfiedIcon from '@mui/icons-material/SentimentVerySatisfied';

const StyledRating = styled(Rating)(({ theme }) => ({
  '& .MuiRating-iconEmpty .MuiSvgIcon-root': {
    color: theme.palette.action.disabled,
  },
}));

const customIcons: {
  [index: string]: {
    icon: React.ReactElement<unknown>;
    label: string;
  };
} = {
  1: {
    icon: <SentimentVeryDissatisfiedIcon color="error" />,
    label: 'Very Dissatisfied',
  },
  2: {
    icon: <SentimentDissatisfiedIcon color="error" />,
    label: 'Dissatisfied',
  },
  3: {
    icon: <SentimentSatisfiedIcon color="warning" />,
    label: 'Neutral',
  },
  4: {
    icon: <SentimentSatisfiedAltIcon color="success" />,
    label: 'Satisfied',
  },
```

Accessibility

(WAI tutorial ↗)

The accessibility of this component relies on:

- A radio group with its fields visually hidden. It contains six radio buttons, one for each star, and another for 0 stars that is checked by default. Be sure to provide a value for the `name` prop that is unique to the parent form.
- Labels for the radio buttons containing actual text ("1 Star", "2 Stars", ...). Be sure to provide a suitable function to the `getLabelText` prop when the page is in a language other than English. You can use the included locales, or provide your own.
- A visually distinct appearance for the rating icons. By default, the rating component uses both a difference of color and shape (filled and empty icons) to indicate the value. In the event that you are using color as the only means to indicate the value, the information should also be also displayed as text, as in this demo. This is important to match success Criterion 1.4.1 ↗ of WCAG2.1.

Edit in Chat JS TS Collapse code

```tsx
import Box from '@mui/material/Box';
import Rating from '@mui/material/Rating';
import StarIcon from '@mui/icons-material/Star';

const labels: { [index: string]: string } = {
  0.5: 'Useless',
  1: 'Useless+',
  1.5: 'Poor',
  2: 'Poor+',
  2.5: 'Ok',
  3: 'Ok+',
  3.5: 'Good',
  4: 'Good+',
  4.5: 'Excellent',
  5: 'Excellent+',
};

export default function TextRating() {
  const value = 3.5;

  return (
    <Box sx={{ width: 200, display: 'flex', alignItems: 'center' }}>
      <Rating
        name="text-feedback"
        value={value}
        readOnly
        precision={0.5}
        emptyIcon={<StarIcon style={{ opacity: 0.55 }} fontSize="inherit" />}
      />
      <Box sx={{ ml: 2 }}>{labels[value]}</Box>
    </Box>
  );
}
```

ARIA

The read only rating has a role of "img", and an aria-label that describes the displayed rating.

Keyboard

Because the rating component uses radio buttons, keyboard interaction follows the native browser behavior. Tab will focus the current rating, and cursor keys control the selected rating.

The read only rating is not focusable.

Testing

When testing the Rating component in environments such as Jest with jsdom, certain user interactions—especially hover-based interactions—may not behave as expected. This is because the component relies on `getBoundingClientRect()` to calculate the position of each icon and determine which icon is currently being hovered. In jsdom, `getBoundingClientRect()` returns `0` values by default, which can lead to incorrect behavior such as `NaN` being passed to the `onChange` handler.

To avoid this issue in your test suite:

- Prefer `fireEvent` over `userEvent` when simulating click events.
- Avoid relying on hover behavior to trigger changes.
- If needed, mock `getBoundingClientRect()` manually for more advanced interactions.

```
// @vitest-environment jsdom

import { Rating } from '@mui/material';
import { render, fireEvent, screen } from '@testing-library/react';

import { describe, test, vi } from 'vitest';

describe('Rating', () => {
  test('should update rating on click', () => {
    const handleChange = vi.fn();
    render(<Rating onChange={(_, newValue) => handleChange(newValue)} />);

    fireEvent.click(screen.getByLabelText('2 Stars'));

    expect(handleChange).toHaveBeenCalledWith(2);
  });
});
```

Copy

API

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

- `<Rating />`