



Search...

Ctrl+K



# Checkbox

Checkboxes allow the user to select one or more items from a set.



Design and Development tips in your inbox. Every weekday.

ads via Carbon

Checkboxes can be used to turn an option on or off.

If you have multiple options appearing in a list, you can preserve space by using checkboxes instead of on/off switches. If you have a single option, avoid using a checkbox and use an on/off switch instead.

[View as Markdown](#)

[Feedback](#)

[Bundle size](#)

[Source](#)

[WAI-ARIA](#)

[Material Design](#)

[Figma](#)

[Sketch](#)

## Basic checkboxes



[Collapse code](#)



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

const label = { slotProps: { input: { 'aria-label': 'Checkbox demo' } } };

export default function Checkboxes() {
  return (
    <div>
      <Checkbox {...label} defaultChecked />
      <Checkbox {...label} />
      <Checkbox {...label} disabled />
      <Checkbox {...label} disabled checked />
    </div>
  );
}
```

# Label

+

You can provide a label to the `Checkbox` thanks to the `FormControlLabel` component.

- Label
- Required \*
- Disabled

[Edit in Chat](#)

JS

TS

[Collapse code](#)

```
import FormGroup from '@mui/material/FormGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import Checkbox from '@mui/material/Checkbox';

export default function CheckboxLabels() {
  return (
    <FormGroup>
      <FormControlLabel control={<Checkbox defaultChecked />} label="Label" />
      <FormControlLabel required control={<Checkbox />} label="Required" />
      <FormControlLabel disabled control={<Checkbox />} label="Disabled" />
    </FormGroup>
  );
}
```

# Size

+

[Edit in Chat](#)

JS

TS

[Collapse code](#)

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

const label = { slotProps: { input: { 'aria-label': 'Checkbox demo' } } };

export default function SizeCheckboxes() {
  return (
    <div>
      <Checkbox {...label} defaultChecked size="small" />
      <Checkbox {...label} defaultChecked />
      <Checkbox {...label} defaultChecked />
    </div>
  );
}
```

```
    {...label}
    defaultChecked
    sx={{ '& .MuiSvgIcon-root': { fontSize: 28 } }}}
  />
</div>
);
}
```

## Color



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

[Collapse code](#)



```
import { pink } from '@mui/material/colors';
import Checkbox from '@mui/material/Checkbox';

const label = { slotProps: { input: { 'aria-label': 'Checkbox demo' } } };

export default function ColorCheckboxes() {
  return (
    <div>
      <Checkbox {...label} defaultChecked />
      <Checkbox {...label} defaultChecked color="secondary" />
      <Checkbox {...label} defaultChecked color="success" />
      <Checkbox {...label} defaultChecked color="default" />
      <Checkbox
        {...label}
        defaultChecked
        sx={{
          color: pink[800],
          '&.Mui-checked': {
            color: pink[600],
          },
        }}
      />
    </div>
  );
}
```

## Icon





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

[Collapse code](#)



```
import Checkbox from '@mui/material/Checkbox';
import FavoriteBorder from '@mui/icons-material/FavoriteBorder';
import Favorite from '@mui/icons-material/Favorite';
import BookmarkBorderIcon from '@mui/icons-material/BookmarkBorder';
import BookmarkIcon from '@mui/icons-material/Bookmark';

const label = { slotProps: { input: { 'aria-label': 'Checkbox demo' } } };

export default function IconCheckboxes() {
  return (
    <div>
      <Checkbox {...label} icon={<FavoriteBorder />} checkedIcon={<Favorite />} />
      <Checkbox
        {...label}
        icon={<BookmarkBorderIcon />}
        checkedIcon={<BookmarkIcon />}
      />
    </div>
  );
}
```

## Controlled



You can control the checkbox with the `checked` and `onChange` props:



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

[Collapse code](#)



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

export default function ControlledCheckbox() {
  const [checked, setChecked] = React.useState(true);

  const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setChecked(event.target.checked);
  };

  return (
    <div>
      <input checked={checked} type="checkbox" />
      <span>{checked ? 'checked' : 'unchecked'}</span>
    </div>
  );
}
```

```
<Checkbox
  checked={checked}
  onChange={handleChange}
  slotProps={{
    input: { 'aria-label': 'controlled' },
  }}
/>
);
}
```

## Indeterminate

+

A checkbox input can only have two states in a form: checked or unchecked. It either submits its value or doesn't. Visually, there are **three** states a checkbox can be in: checked, unchecked, or indeterminate.

You can change the indeterminate icon using the `indeterminateIcon` prop.

Parent

Child 1

Child 2

[Edit in Chat](#)

JS

TS

[Collapse code](#)



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

export default function IndeterminateCheckbox() {
  const [checked, setChecked] = React.useState([true, false]);

  const handleChange1 = (event: React.ChangeEvent<HTMLInputElement>) => {
    setChecked([event.target.checked, event.target.checked]);
  };

  const handleChange2 = (event: React.ChangeEvent<HTMLInputElement>) => {
    setChecked([event.target.checked, checked[1]]);
  };

  const handleChange3 = (event: React.ChangeEvent<HTMLInputElement>) => {
    setChecked([checked[0], event.target.checked]);
  };

  const children = (
    <Box sx={{ display: 'flex', flexDirection: 'column', ml: 3 }}>
      <FormControlLabel
```

```

    label="Child 1"
    control={<Checkbox checked={checked[0]} onChange={handleChange2} />}
  />
<FormControlLabel
  label="Child 2"
  control={<Checkbox checked={checked[1]} onChange={handleChange3} />}
/>
</Box>
);

return (
<div>
<FormControlLabel
  label="Parent"
  control={


```

⚠ When indeterminate is set, the value of the `checked` prop only impacts the form submitted values. It has no accessibility or UX implications.

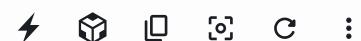
## FormGroup

`FormGroup` is a helpful wrapper used to group selection control components.

<p>Assign responsibility</p> <p><input checked="" type="checkbox"/> Gilad Gray</p> <p><input type="checkbox"/> Jason Killian</p> <p><input type="checkbox"/> Antoine Llorca</p> <p>Be careful</p>	<p>Pick two *</p> <p><input checked="" type="checkbox"/> Gilad Gray</p> <p><input type="checkbox"/> Jason Killian</p> <p><input type="checkbox"/> Antoine Llorca</p> <p>You can display an error</p>
---	--

★ [Edit in Chat](#) JS TS

[Hide code](#)



```

import * as React from 'react';
import Box from '@mui/material/Box';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import FormGroup from '@mui/material/FormGroup';
import FormHelperText from '@mui/material/FormHelperText';
import Checkbox from '@mui/material/Checkbox';

export default function CheckboxesGroup() {

```

```

const [state, setState] = React.useState({
  gilad: true,
  jason: false,
  antoine: false,
});

const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
  setState({
    ...state,
    [event.target.name]: event.target.checked,
  });
};

const { gilad, jason, antoine } = state;
const error = [gilad, jason, antoine].filter((v) => v).length !== 2;

return (
  <Box sx={{ display: 'flex' }}>
    <FormControl sx={{ m: 3 }} component="fieldset" variant="standard">
      <FormLabel component="legend">Assign responsibility</FormLabel>
      <FormGroup>
        <FormControlLabel
          control={
            <Checkbox checked={gilad} onChange={handleChange} name="gilad" />
          }
          label="Gilad Gray"
        />
      
```

## Label placement

You can change the placement of the label:

Label placement



[Edit in Chat](#)

JS TS

[Hide code](#)



```

import Checkbox from '@mui/material/Checkbox';
import FormGroup from '@mui/material/FormGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import FormLabel from '@mui/material/FormLabel';

export default function FormControlLabelPosition() {
  return (
    <FormControl component="fieldset">
      <FormLabel component="legend">Label placement</FormLabel>

```

```
<FormGroup aria-label="position" row>
  <FormControlLabel
    value="bottom"
    control={<Checkbox />}
    label="Bottom"
    labelPlacement="bottom"
  />
  <FormControlLabel
    value="end"
    control={<Checkbox />}
    label="End"
    labelPlacement="end"
  />
</FormGroup>
</FormControl>
);
}
```

## 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](#)

[Collapse code](#)



```
import { styled } from '@mui/material/styles';
import Checkbox, { CheckboxProps } from '@mui/material/Checkbox';

const BpIcon = styled('span')(({ theme }) => ({
  borderRadius: 3,
  width: 16,
  height: 16,
  boxShadow: 'inset 0 0 0 1px rgba(16,22,26,.2), inset 0 -1px 0 rgba(16,22,26,.1)',
  backgroundColor: '#f5f5f8fa',
  backgroundImage: 'linear-gradient(180deg,hsla(0,0%,100%,.8),hsla(0,0%,100%,0))',
  '.Mui-focusVisible &': {
    outline: '2px auto rgba(19,124,189,.6)',
    outlineOffset: 2,
  },
  '&:hover': {
    backgroundColor: '#ebf1f5',
    ...theme.applyStyles('dark', {
      backgroundColor: '#30404d',
    }),
  },
  '&:disabled': {
    color: 'gray',
  }
});
```

```
boxShadow: 'none',
background: 'rgba(206,217,224,.5)',
...theme.applyStyles('dark', {
  background: 'rgba(57,75,89,.5)',
}),
},
...theme.applyStyles('dark', {
  boxShadow: '0 0 0 1px rgb(16 22 26 / 40%)',
  backgroundColor: '#394b59',
  backgroundImage: 'linear-gradient(180deg,hsla(0,0%,100%,.05),hsla(0,0%,100%,0))',
}),
));
});
```

```
const BpCheckedIcon = styled(BpIcon)({
  backgroundColor: '#137cbd',
  backgroundImage: 'linear-gradient(180deg,hsla(0,0%,100%,.1),hsla(0,0%,100%,0))',
  border: '1px solid #137cbd',
  color: '#137cbd',
  width: 16,
  height: 16,
  '&:hover': {
    border: '1px solid #137cbd',
    color: '#137cbd',
  },
});
```

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

## When to use

- [Checkboxes vs. Radio Buttons](#)
- [Checkboxes vs. Switches](#)

## Accessibility

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

- All form controls should have labels, and this includes radio buttons, checkboxes, and switches. In most cases, this is done by using the `<label>` element ([FormControlLabel](#)).
- When a label can't be used, it's necessary to add an attribute directly to the input component. In this case, you can apply the additional attribute (for example `aria-label`, `aria-labelledby`, `title`) via the `slotProps.input` prop.

```
<Checkbox
  value="checkedA"
  slotProps={{
    input: { 'aria-label': 'Checkbox A' },
  }}
/>
```

Copy

## API

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

- [`<Checkbox />`](#)
- [`<FormControl />`](#)
- [`<FormControlLabel />`](#)
- [`<FormGroup />`](#)
- [`<FormLabel />`](#)

 Edit this page

Was this page helpful?  

[« Button Group](#)

[Floating Action Button »](#)