Radio Group

The Radio Group allows the user to select one option from a set.

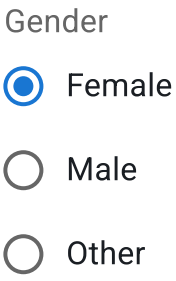
Use radio buttons when the user needs to see all available options. If available options can be collapsed, consider using a Select component because it uses less space.

Radio buttons should have the most commonly used option selected by default.

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

Radio group

`RadioGroup` is a helpful wrapper used to group `Radio` components that provides an easier API, and proper keyboard accessibility to the group.



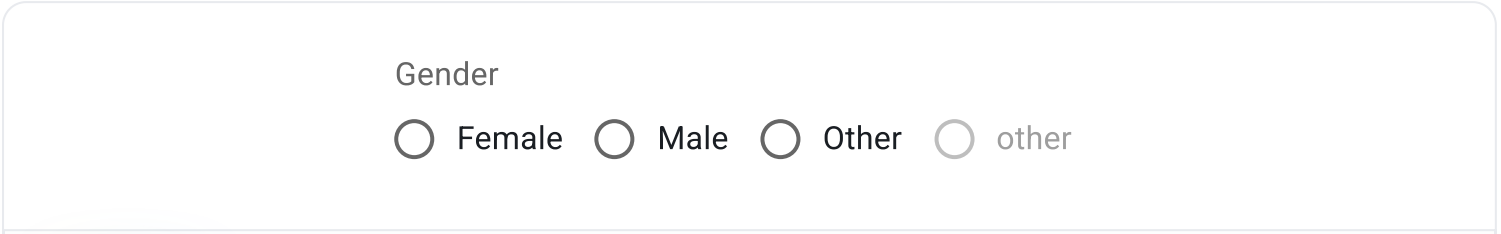
Edit in Chat | JS | TS | Collapse code

```
import Radio from '@mui/material/Radio';
import RadioGroup from '@mui/material/RadioGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import FormLabel from '@mui/material/FormLabel';

export default function RadioButtonsGroup() {
  return (
    <FormControl>
```

```
      <FormLabel id="demo-radio-buttons-group-label">Gender</FormLabel>
      <RadioGroup
        aria-labelledby="demo-radio-buttons-group-label"
        defaultValue="female"
        name="radio-buttons-group"
      >
        <FormControlLabel value="female" control={<Radio />} label="Female" />
        <FormControlLabel value="male" control={<Radio />} label="Male" />
        <FormControlLabel value="other" control={<Radio />} label="Other" />
      </RadioGroup>
    </FormControl>
  );
}
```

Direction

To lay out the buttons horizontally, set the `row` prop:



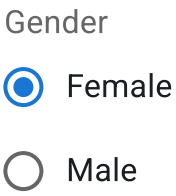
Edit in Chat | JS | TS | Hide code

```
import Radio from '@mui/material/Radio';
import RadioGroup from '@mui/material/RadioGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import FormLabel from '@mui/material/FormLabel';

export default function RowRadioButtonsGroup() {
  return (
    <FormControl>
      <FormLabel id="demo-row-radio-buttons-group-label">Gender</FormLabel>
      <RadioGroup
        row
        aria-labelledby="demo-row-radio-buttons-group-label"
        name="row-radio-buttons-group"
      >
        <FormControlLabel value="female" control={<Radio />} label="Female" />
        <FormControlLabel value="male" control={<Radio />} label="Male" />
        <FormControlLabel value="other" control={<Radio />} label="Other" />
        <FormControlLabel
          value="disabled"
          disabled
          control={<Radio />}
          label="other"
        />
      </RadioGroup>
    </FormControl>
```

```
  );
}
```

Controlled

You can control the radio with the `value` and `onChange` props:



Edit in Chat | JS | TS | Collapse code

```tsx
import * as React from 'react';
import Radio from '@mui/material/Radio';
import RadioGroup from '@mui/material/RadioGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import FormLabel from '@mui/material/FormLabel';

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

  const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setValue((event.target as HTMLInputElement).value);
  };

  return (
    <FormControl>
      <FormLabel id="demo-controlled-radio-buttons-group">Gender</FormLabel>
      <RadioGroup
        aria-labelledby="demo-controlled-radio-buttons-group"
        name="controlled-radio-buttons-group"
        value={value}
        onChange={handleChange}
      >
        <FormControlLabel value="female" control={<Radio />} label="Female" />
        <FormControlLabel value="male" control={<Radio />} label="Male" />
      </RadioGroup>
    </FormControl>
  );
}
```

Standalone radio buttons

`Radio` can also be used standalone, without the RadioGroup wrapper.

Edit in Chat | JS | TS | Collapse code

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

export default function RadioButtons() {
  const [selectedValue, setSelectedValue] = React.useState('a');

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

  return (
    <div>
      <Radio
        checked={selectedValue === 'a'}
        onChange={handleChange}
        value="a"
        name="radio-buttons"
        inputProps={{ 'aria-label': 'A' }}
      />
      <Radio
        checked={selectedValue === 'b'}
        onChange={handleChange}
        value="b"
        name="radio-buttons"
        inputProps={{ 'aria-label': 'B' }}
      />
    </div>
  );
}
```

Size

Use the `size` prop or customize the font size of the svg icons to change the size of the radios.

Edit in Chat JS TS Collapse code

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

export default function SizeRadioButtons() {
  const [selectedValue, setSelectedValue] = React.useState('a');
  const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setSelectedValue(event.target.value);
  };

  const controlProps = (item: string) => ({
    checked: selectedValue === item,
    onChange: handleChange,
    value: item,
    name: 'size-radio-button-demo',
    inputProps: { 'aria-label': item },
  });

  return (
    <div>
      <Radio {...controlProps('a')} size="small" />
      <Radio {...controlProps('b')} />
      <Radio
        {...controlProps('c')}
        sx={{
          '& .MuiSvgIcon-root': {
            fontSize: 28,
          },
        }}
      />
    </div>
  );
}
```

Color

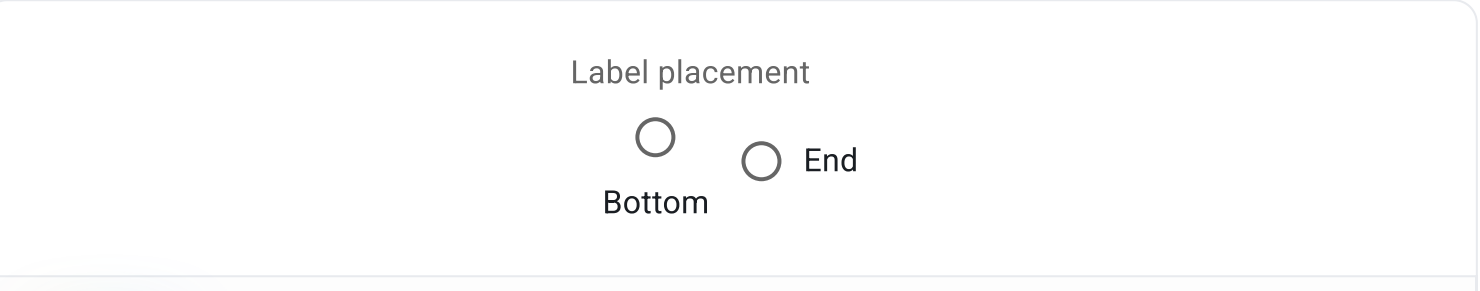
Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import { pink } from '@mui/material/colors';
import Radio from '@mui/material/Radio';

export default function ColorRadioButtons() {
  const [selectedValue, setSelectedValue] = React.useState('a');

  const handleChange = (event: React.ChangeEvent<HTMLInputElement>) => {
```

```
    setSelectedValue(event.target.value);
  };

  const controlProps = (item: string) => ({
    checked: selectedValue === item,
    onChange: handleChange,
    value: item,
    name: 'color-radio-button-demo',
    inputProps: { 'aria-label': item },
  });

  return (
    <div>
      <Radio {...controlProps('a')} />
      <Radio {...controlProps('b')} color="secondary" />
      <Radio {...controlProps('c')} color="success" />
      <Radio {...controlProps('d')} color="default" />
      <Radio
        {...controlProps('e')}
        sx={{
          color: pink[800],
          '&.Mui-checked': {
            color: pink[600],
          },
        }}
      />
    </div>
  );
}
```

Label placement

You can change the placement of the label with the `FormControlLabel` component's `labelPlacement` prop:



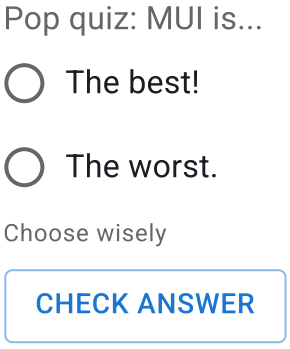
Edit in Chat JS TS Hide code

```
import Radio from '@mui/material/Radio';
import RadioGroup from '@mui/material/RadioGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import FormLabel from '@mui/material/FormLabel';

export default function FormControlLabelPlacement() {
  return (
```

```
    <FormControl>
      <FormLabel id="demo-form-control-label-placement">Label placement</FormLabel>
      <RadioGroup
        row
        aria-labelledby="demo-form-control-label-placement"
        name="position"
        defaultValue="top"
      >
        <FormControlLabel
          value="bottom"
          control={<Radio />}
          label="Bottom"
          labelPlacement="bottom"
        />
        <FormControlLabel value="end" control={<Radio />} label="End" />
      </RadioGroup>
    </FormControl>
  );
}
```

Show error

In general, radio buttons should have a value selected by default. If this is not the case, you can display an error if no value is selected when the form is submitted:



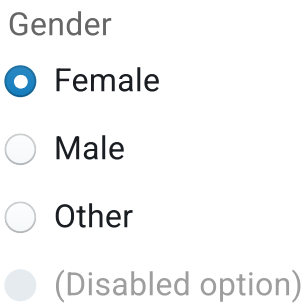
Edit in Chat | JS | TS | Hide code

```
import * as React from 'react';
import Radio from '@mui/material/Radio';
import RadioGroup from '@mui/material/RadioGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
import FormHelperText from '@mui/material/FormHelperText';
import FormLabel from '@mui/material/FormLabel';
import Button from '@mui/material/Button';

export default function ErrorRadios() {
```

```tsx
  const [value, setValue] = React.useState('');
  const [error, setError] = React.useState(false);
  const [helperText, setHelperText] = React.useState('Choose wisely');

  const handleRadioChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setValue((event.target as HTMLInputElement).value);
    setHelperText(' ');
    setError(false);
  };

  const handleSubmit = (event: React.FormEvent<HTMLFormElement>) => {
    event.preventDefault();

    if (value === 'best') {
      setHelperText('You got it!');
      setError(false);
    } else if (value === 'worst') {
      setHelperText('Sorry, wrong answer!');
      setError(true);
    } else {
      setHelperText('Please select an option.');
      setError(true);
    }
  };

  return (
    <form onSubmit={handleSubmit}>
      <FormControl sx={{ m: 3 }} error={error} variant="standard">
```

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 { styled } from '@mui/material/styles';
import Radio, { RadioProps } from '@mui/material/Radio';
import RadioGroup from '@mui/material/RadioGroup';
import FormControlLabel from '@mui/material/FormControlLabel';
import FormControl from '@mui/material/FormControl';
```

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

const BpIcon = styled('span')(({ theme }) => ({
  borderRadius: '50%',
  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: '#f5f8fa',
  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,
  },
  'input:hover ~ &': {
    backgroundColor: '#ebf1f5',
    ...theme.applyStyles('dark', {
      backgroundColor: '#30404d',
    }),
  },
  'input:disabled ~ &': {
    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))',
  }),
}));
```

`useRadioGroup`

For advanced customization use cases, a `useRadioGroup()` hook is exposed. It returns the context value of the parent radio group. The Radio component uses this hook internally.

API

```
import { useRadioGroup } from '@mui/material/RadioGroup';
```

Copy

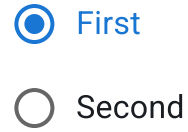
Returns

`value` (*object*):

- `value.name` (*string* [optional]): The name used to reference the value of the control.
- `value.onChange` (*func* [optional]): Callback fired when a radio button is selected.

- `value.value` (*any* [optional]): Value of the selected radio button.

Example



Edit in Chat JS TS Collapse code

```tsx
import { styled } from '@mui/material/styles';
import RadioGroup, { useRadioGroup } from '@mui/material/RadioGroup';
import FormControlLabel, {
  FormControlLabelProps,
} from '@mui/material/FormControlLabel';
import Radio from '@mui/material/Radio';

interface StyledFormControlLabelProps extends FormControlLabelProps {
  checked: boolean;
}

const StyledFormControlLabel = styled((props: StyledFormControlLabelProps) => (
  <FormControlLabel {...props} />
))(({ theme }) => ({
  variants: [
    {
      props: { checked: true },
      style: {
        '.MuiFormControlLabel-label': {
          color: theme.palette.primary.main,
        },
      },
    },
  ],
}));

function MyFormControlLabel(props: FormControlLabelProps) {
  const radioGroup = useRadioGroup();

  let checked = false;

  if (radioGroup) {
    checked = radioGroup.value === props.value;
  }

  return <StyledFormControlLabel checked={checked} {...props} />;
}
```

When to use

- Checkboxes vs. Radio Buttons

Accessibility

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

- 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 `inputProps` property.

```
<Radio
  value="radioA"
  inputProps={{
    'aria-label': 'Radio A',
  }}
/>
```

Copy

API

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

- `<FormControl />`
- `<FormControlLabel />`
- `<FormLabel />`
- `<Radio />`
- `<RadioGroup />`