Progress

Progress indicators commonly known as spinners, express an unspecified wait time or display the length of a process.

Progress indicators inform users about the status of ongoing processes, such as loading an app, submitting a form, or saving updates.

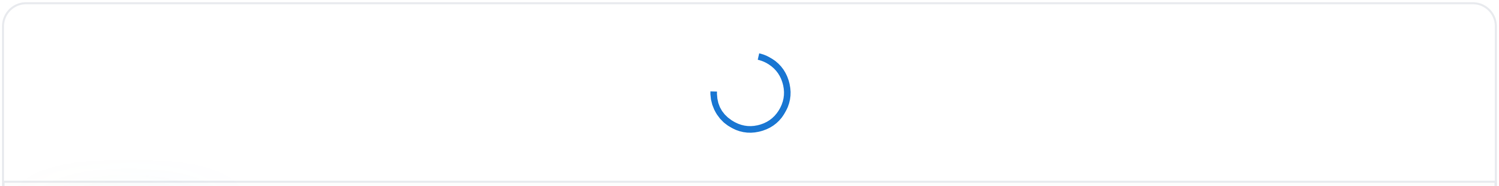
- **Determinate** indicators display how long an operation will take.
- **Indeterminate** indicators visualize an unspecified wait time.

The animations of the components rely on CSS as much as possible to work even before the JavaScript is loaded.

View as Markdown Feedback Bundle size Source Material Design Figma Sketch

Circular

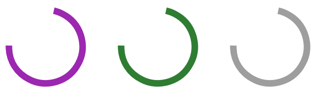
Circular indeterminate



Edit in Chat JS TS Collapse code

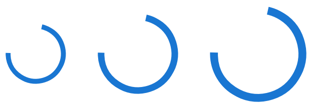
```
import CircularProgress from '@mui/material/CircularProgress';
import Box from '@mui/material/Box';

export default function CircularIndeterminate() {
  return (
    <Box sx={{ display: 'flex' }}>
      <CircularProgress />
    </Box>
  );
}
```

Circular color



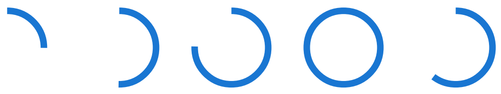
```tsx
import Stack from '@mui/material/Stack';
import CircularProgress from '@mui/material/CircularProgress';

export default function CircularColor() {
  return (
    <Stack sx={{ color: 'grey.500' }} spacing={2} direction="row">
      <CircularProgress color="secondary" />
      <CircularProgress color="success" />
      <CircularProgress color="inherit" />
    </Stack>
  );
}
```

Circular size



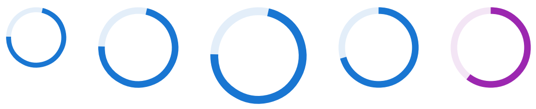
```tsx
import Stack from '@mui/material/Stack';
import CircularProgress from '@mui/material/CircularProgress';

export default function CircularSize() {
  return (
    <Stack spacing={2} direction="row" alignItems="center">
      <CircularProgress size="30px" />
      <CircularProgress size={40} />
      <CircularProgress size="3rem" />
    </Stack>
  );
}
```

Circular determinate



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

export default function CircularDeterminate() {
  const [progress, setProgress] = React.useState(0);

  React.useEffect(() => {
    const timer = setInterval(() => {
      setProgress((prevProgress) => (prevProgress >= 100 ? 0 : prevProgress + 10));
    }, 800);

    return () => {
      clearInterval(timer);
    };
  }, []);

  return (
    <Stack spacing={2} direction="row">
      <CircularProgress variant="determinate" value={25} />
      <CircularProgress variant="determinate" value={50} />
      <CircularProgress variant="determinate" value={75} />
      <CircularProgress variant="determinate" value={100} />
      <CircularProgress variant="determinate" value={progress} />
    </Stack>
  );
}
```

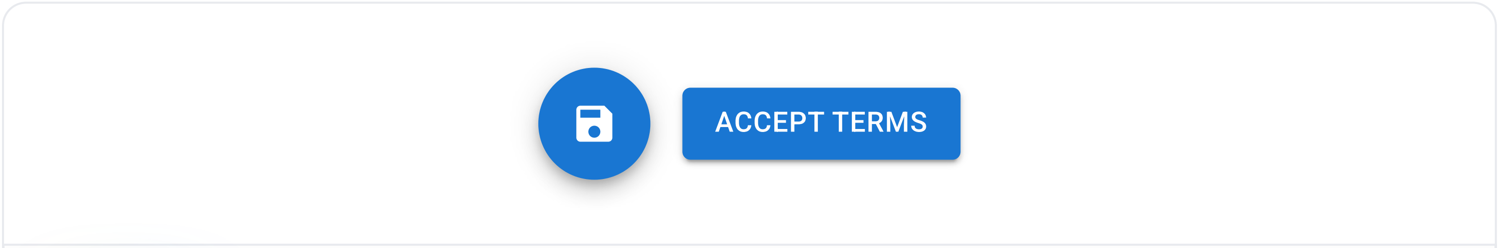
Circular track



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

```
export default function CircularEnableTrack() {
  const [progress, setProgress] = React.useState(0);

  React.useEffect(() => {
    const timer = setInterval(() => {
      setProgress((prevProgress) => (prevProgress >= 100 ? 0 : prevProgress + 10));
    }, 800);

    return () => {
      clearInterval(timer);
    };
  }, []);

  return (
    <Stack spacing={2} direction="row">
      <CircularProgress enableTrackSlot size="30px" />
      <CircularProgress enableTrackSlot size={40} />
      <CircularProgress enableTrackSlot size="3rem" />
      <CircularProgress enableTrackSlot variant="determinate" value={70} />
      <CircularProgress
        enableTrackSlot
        variant="determinate"
        color="secondary"
        value={progress}
      />
    </Stack>
  );
}
```

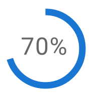
Interactive integration



Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import CircularProgress from '@mui/material/CircularProgress';
import { green } from '@mui/material/colors';
import Button from '@mui/material/Button';
import Fab from '@mui/material/Fab';
import CheckIcon from '@mui/icons-material/Check';
import SaveIcon from '@mui/icons-material/Save';

export default function CircularIntegration() {
  const [loading, setLoading] = React.useState(false);
  const [success, setSuccess] = React.useState(false);
  const timer = React.useRef<ReturnType<typeof setTimeout>>(undefined);
  const buttonSx = {
    ...(success && {
      bgcolor: green[500],
      '&:hover': {
        bgcolor: green[700],
      },
    }),
  };

  React.useEffect(() => {
    return () => {
      clearTimeout(timer.current);
    };
  }, []);

  const handleButtonClick = () => {
    if (!loading) {
      setSuccess(false);
      setLoading(true);
      timer.current = setTimeout(() => {
        setSuccess(true);
        setLoading(false);
      }, 2000);
    }
```

Circular with label

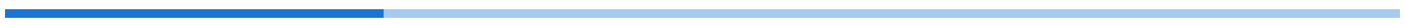


Edit in Chat JS TS Collapse code

```
import * as React from 'react';
import CircularProgress, {
  CircularProgressProps,
} from '@mui/material/CircularProgress';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';

function CircularProgressWithLabel(
  props: CircularProgressProps & { value: number },
) {
  return (
    <Box sx={{ position: 'relative', display: 'inline-flex' }}>
      <CircularProgress variant="determinate" {...props} />
      <Box
        sx={{
          top: 0,
          left: 0,
          bottom: 0,
          right: 0,
          position: 'absolute',
          display: 'flex',
          alignItems: 'center',
          justifyContent: 'center',
        }}
      >
        <Typography
          variant="caption"
          component="div"
          sx={{ color: 'text.secondary' }}
        >{`${Math.round(props.value)}%`}</Typography>
      </Box>
    </Box>
  );
}

export default function CircularWithValueLabel() {
  const [progress, setProgress] = React.useState(10);
```

Linear

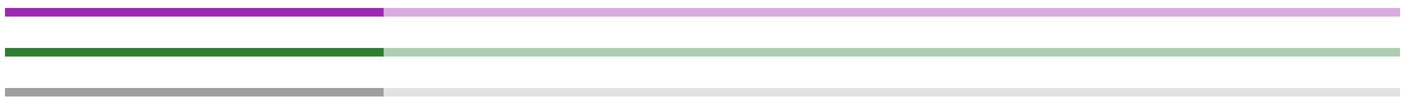
Linear indeterminate



Edit in Chat JS TS Collapse code

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

export default function LinearIndeterminate() {
  return (
    <Box sx={{ width: '100%' }}>
      <LinearProgress />
    </Box>
  );
}
```

Linear color



Edit in Chat JS TS Collapse code

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

export default function LinearColor() {
  return (
    <Stack sx={{ width: '100%', color: 'grey.500' }} spacing={2}>
      <LinearProgress color="secondary" />
      <LinearProgress color="success" />
      <LinearProgress color="inherit" />
    </Stack>
  );
}
```

Linear determinate

Edit in Chat JS TS Collapse code

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

export default function LinearDeterminate() {
  const [progress, setProgress] = React.useState(0);

  React.useEffect(() => {
    const timer = setInterval(() => {
      setProgress((oldProgress) => {
        if (oldProgress === 100) {
          return 0;
        }
        const diff = Math.random() * 10;
        return Math.min(oldProgress + diff, 100);
      });
    }, 500);

    return () => {
      clearInterval(timer);
    };
  }, []);

  return (
    <Box sx={{ width: '100%' }}>
      <LinearProgress variant="determinate" value={progress} />
    </Box>
  );
}
```

Linear buffer

Edit in Chat JS TS Collapse code

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

export default function LinearBuffer() {
  const [progress, setProgress] = React.useState(0);
  const [buffer, setBuffer] = React.useState(10);

  const progressRef = React.useRef(() => {});
  React.useEffect(() => {
    progressRef.current = () => {
      if (progress === 100) {
        setProgress(0);
        setBuffer(10);
      } else {
        setProgress(progress + 1);
        if (buffer < 100 && progress % 5 === 0) {
          const newBuffer = buffer + 1 + Math.random() * 10;
          setBuffer(newBuffer > 100 ? 100 : newBuffer);
        }
      }
    };
  });

  React.useEffect(() => {
    const timer = setInterval(() => {
      progressRef.current();
    }, 100);

    return () => {
      clearInterval(timer);
    };
  }, []);

  return (
    <Box sx={{ width: '100%' }}>
      <LinearProgress variant="buffer" value={progress} valueBuffer={buffer} />
    </Box>
```

Linear with label

Edit in Chat JS TS Collapse code

```tsx
import * as React from 'react';
import LinearProgress, { LinearProgressProps } from '@mui/material/LinearProgress';
import Typography from '@mui/material/Typography';
import Box from '@mui/material/Box';

function LinearProgressWithLabel(props: LinearProgressProps & { value: number }) {
  return (
    <Box sx={{ display: 'flex', alignItems: 'center' }}>
      <Box sx={{ width: '100%', mr: 1 }}>
        <LinearProgress variant="determinate" {...props} />
      </Box>
      <Box sx={{ minWidth: 35 }}>
        <Typography
          variant="body2"
          sx={{ color: 'text.secondary' }}
        >{`${Math.round(props.value)}%`}</Typography>
      </Box>
    </Box>
  );
}

export default function LinearWithValueLabel() {
  const [progress, setProgress] = React.useState(10);

  React.useEffect(() => {
    const timer = setInterval(() => {
      setProgress((prevProgress) => (prevProgress >= 100 ? 10 : prevProgress + 10));
    }, 800);
    return () => {
      clearInterval(timer);
    };
  }, []);

  return (
    <Box sx={{ width: '100%' }}>
      <LinearProgressWithLabel value={progress} />
    </Box>
  );
```

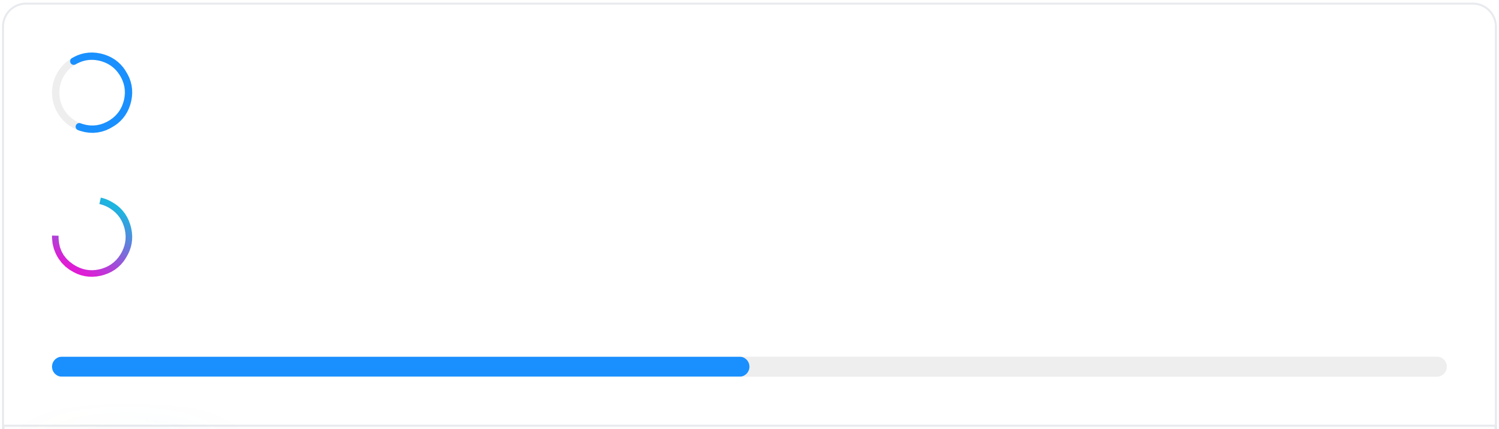
Non-standard ranges

The progress components accept a value in the range 0 - 100. This simplifies things for screen-reader users, where these are the default min / max values. Sometimes, however, you might be working with a data source where the values fall outside this range. Here's how you can easily transform a value in any range to a scale of 0 - 100:

```
// MIN = Minimum expected value
// MAX = Maximum expected value
// Function to normalise the values (MIN / MAX could be integrated)
const normalise = (value) => ((value - MIN) * 100) / (MAX - MIN);

// Example component that utilizes the `normalise` function at the point of render.
function Progress(props) {
  return (
    <React.Fragment>
      <CircularProgress variant="determinate" value={normalise(props.value)} />
      <LinearProgress variant="determinate" value={normalise(props.value)} />
    </React.Fragment>
  );
}
```

Customization

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



Edit in Chat JS TS Hide code

```
import * as React from 'react';
import { styled } from '@mui/material/styles';
import Stack from '@mui/material/Stack';
import CircularProgress, {
  circularProgressClasses,
  CircularProgressProps,
} from '@mui/material/CircularProgress';
import LinearProgress, { linearProgressClasses } from '@mui/material/LinearProgress';

const BorderLinearProgress = styled(LinearProgress)(({ theme }) => ({
  height: 10,
  borderRadius: 5,
  [`&.${linearProgressClasses.colorPrimary}`]: {
    backgroundColor: theme.palette.grey[200],
    ...theme.applyStyles('dark', {
      backgroundColor: theme.palette.grey[800],
    }),
  },
  [`& .${linearProgressClasses.bar}`]: {
    borderRadius: 5,
    backgroundColor: '#1a90ff',
    ...theme.applyStyles('dark', {
      backgroundColor: '#308fe8',
    }),
  },
}));

// Inspired by the former Facebook spinners.
function FacebookCircularProgress(props: CircularProgressProps) {
  return (
    <CircularProgress
      variant="indeterminate"
      disableShrink
      enableTrackSlot
      sx={(theme) => ({
        color: '#1a90ff',
        animationDuration: '550ms',
```

Delaying appearance

There are 3 important limits to know around response time. The ripple effect of the `ButtonBase` component ensures that the user feels that the UI is reacting instantaneously. Normally, no special feedback is necessary during delays of more than 0.1 but less than 1.0 second. After 1.0 second, you can display a loader to keep user's flow of thought uninterrupted.

LOADING

SIMULATE A LOAD

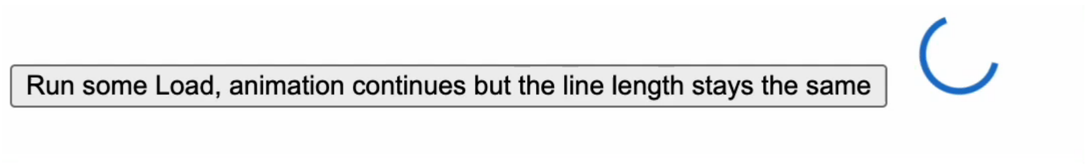
Edit in Chat JS TS Hide code

```
import * as React from 'react';
import Box from '@mui/material/Box';
import Fade from '@mui/material/Fade';
import Button from '@mui/material/Button';
import CircularProgress from '@mui/material/CircularProgress';
import Typography from '@mui/material/Typography';

export default function DelayingAppearance() {
  const [loading, setLoading] = React.useState(false);
  const [query, setQuery] = React.useState('idle');
  const timerRef = React.useRef<ReturnType<typeof setTimeout>>(undefined);

  React.useEffect(
    () => () => {
      clearTimeout(timerRef.current);
    },
    [],
  );

  const handleClickLoading = () => {
    setLoading((prevLoading) => !prevLoading);
  };

  const handleClickQuery = () => {
    if (timerRef.current) {
      clearTimeout(timerRef.current);
    }

    if (query !== 'idle') {
      setQuery('idle');
      return;
    }

    setQuery('progress');
    timerRef.current = setTimeout(() => {
      setQuery('success');
    }, 2000);
  };
```

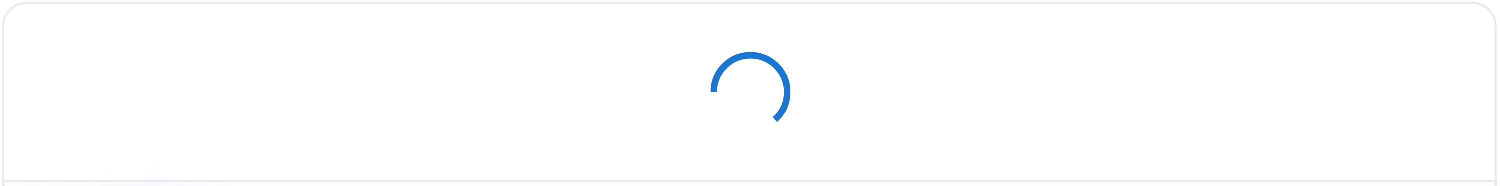
Limitations

High CPU load

Under heavy load, you might lose the stroke dash animation or see random `CircularProgress` ring widths. You should run processor intensive operations in a web worker or by batch in order not to block the main rendering thread.


Run some Load, animation continues but the line length stays the same

When it's not possible, you can leverage the `disableShrink` prop to mitigate the issue. See this issue.



Edit in Chat JS TS Collapse code

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

export default function CircularUnderLoad() {
  return <CircularProgress disableShrink />;
}
```

High frequency updates

The `LinearProgress` uses a transition on the CSS transform property to provide a smooth update between different values. The default transition duration is 200ms. In the event a parent component updates the `value` prop too quickly, you will at least experience a 200ms delay between the re-render and the progress bar fully updated.

If you need to perform 30 re-renders per second or more, we recommend disabling the transition:

```
.MuiLinearProgress-bar {
  transition: none;
}
```

API

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

- `<CircularProgress />`
- `<LinearProgress />`

Edit this page

Was this page helpful?

< Dialog

Skeleton >

MUI • Blog ↗ • Store ↗