Avatar

Avatars are found throughout material design with uses in everything from tables to dialog menus.

View as Markdown | Feedback | Bundle size | Source | Figma | Sketch

Image avatars

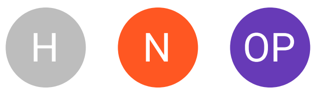
Image avatars can be created by passing standard `img` props `src` or `srcSet` to the component.

Edit in Chat | JS | TS | Collapse code

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

export default function ImageAvatars() {
  return (
    <Stack direction="row" spacing={2}>
      <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
      <Avatar alt="Travis Howard" src="/static/images/avatar/2.jpg" />
      <Avatar alt="Cindy Baker" src="/static/images/avatar/3.jpg" />
    </Stack>
  );
}
```

Letter avatars

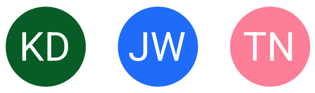
Avatars containing simple characters can be created by passing a string as `children`.



Edit in Chat JS TS Collapse code

```tsx
import Avatar from '@mui/material/Avatar';
import Stack from '@mui/material/Stack';
import { deepOrange, deepPurple } from '@mui/material/colors';

export default function LetterAvatars() {
  return (
    <Stack direction="row" spacing={2}>
      <Avatar>H</Avatar>
      <Avatar sx={{ bgcolor: deepOrange[500] }}>N</Avatar>
      <Avatar sx={{ bgcolor: deepPurple[500] }}>OP</Avatar>
    </Stack>
  );
}
```

You can use different background colors for the avatar. The following demo generates the color based on the name of the person.

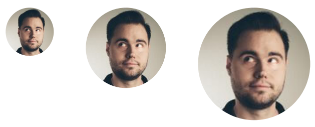


Edit in Chat JS TS Collapse code

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

function stringToColor(string: string) {
  let hash = 0;
  let i;

  /* eslint-disable no-bitwise */
  for (i = 0; i < string.length; i += 1) {
    hash = string.charCodeAt(i) + ((hash << 5) - hash);
  }

  let color = '#';

  for (i = 0; i < 3; i += 1) {
    const value = (hash >> (i * 8)) & 0xff;
    color += `00${value.toString(16)}`.slice(-2);
  }
  /* eslint-enable no-bitwise */
  return color;
}

function stringAvatar(name: string) {
  return {
    sx: {
      bgcolor: stringToColor(name),
    },
    children: `${name.split(' ')[0][0]}${name.split(' ')[1][0]}`,
  };
}

export default function BackgroundLetterAvatars() {
  return (
    <Stack direction="row" spacing={2}>
      <Avatar {...stringAvatar('Kent Dodds')} />
      <Avatar {...stringAvatar('Jed Watson')} />
      <Avatar {...stringAvatar('Tim Neutkens')} />
```

Sizes

You can change the size of the avatar with the `height` and `width` CSS properties.

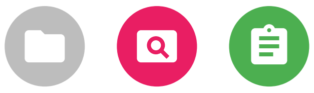


Edit in Chat JS TS Collapse code

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

export default function SizeAvatars() {
  return (
    <Stack direction="row" spacing={2}>
      <Avatar
        alt="Remy Sharp"
        src="/static/images/avatar/1.jpg"
        sx={{ width: 24, height: 24 }}
      />
      <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
      <Avatar
        alt="Remy Sharp"
        src="/static/images/avatar/1.jpg"
        sx={{ width: 56, height: 56 }}
      />
    </Stack>
  );
}
```

Icon avatars

Icon avatars are created by passing an icon as `children`.

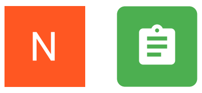


Edit in Chat JS TS Collapse code

```tsx
import { green, pink } from '@mui/material/colors';
import Avatar from '@mui/material/Avatar';
import Stack from '@mui/material/Stack';
import FolderIcon from '@mui/icons-material/Folder';
import PageviewIcon from '@mui/icons-material/Pageview';
import AssignmentIcon from '@mui/icons-material/Assignment';

export default function IconAvatars() {
  return (
    <Stack direction="row" spacing={2}>
      <Avatar>
        <FolderIcon />
      </Avatar>
      <Avatar sx={{ bgcolor: pink[500] }}>
        <PageviewIcon />
      </Avatar>
      <Avatar sx={{ bgcolor: green[500] }}>
        <AssignmentIcon />
      </Avatar>
    </Stack>
  );
}
```

Variants

If you need square or rounded avatars, use the `variant` prop.



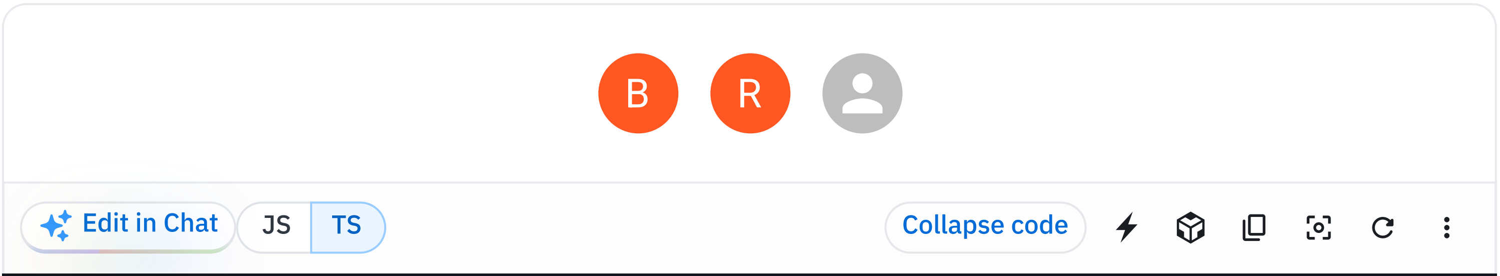
Edit in Chat JS TS Collapse code

```tsx
import Avatar from '@mui/material/Avatar';
import Stack from '@mui/material/Stack';
import { deepOrange, green } from '@mui/material/colors';
import AssignmentIcon from '@mui/icons-material/Assignment';

export default function VariantAvatars() {
  return (
    <Stack direction="row" spacing={2}>
      <Avatar sx={{ bgcolor: deepOrange[500] }} variant="square">
        N
      </Avatar>
      <Avatar sx={{ bgcolor: green[500] }} variant="rounded">
        <AssignmentIcon />
      </Avatar>
    </Stack>
  );
}
```

Fallbacks

If there is an error loading the avatar image, the component falls back to an alternative in the following order:

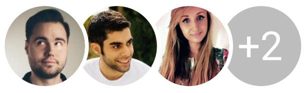
- the provided children
- the first letter of the `alt` text
- a generic avatar icon



```
import Avatar from '@mui/material/Avatar';
import Stack from '@mui/material/Stack';
import { deepOrange } from '@mui/material/colors';

export default function FallbackAvatars() {
  return (
    <Stack direction="row" spacing={2}>
      <Avatar
        sx={{ bgcolor: deepOrange[500] }}
        alt="Remy Sharp"
        src="/broken-image.jpg"
      >
        B
      </Avatar>
      <Avatar
        sx={{ bgcolor: deepOrange[500] }}
        alt="Remy Sharp"
        src="/broken-image.jpg"
      />
      <Avatar src="/broken-image.jpg" />
    </Stack>
  );
}
```

Grouped

`AvatarGroup` renders its children as a stack. Use the `max` prop to limit the number of avatars.

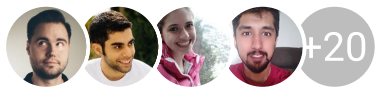


Edit in Chat JS TS Collapse code

```
import Avatar from '@mui/material/Avatar';
import AvatarGroup from '@mui/material/AvatarGroup';

export default function GroupAvatars() {
  return (
    <AvatarGroup max={4}>
      <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
      <Avatar alt="Travis Howard" src="/static/images/avatar/2.jpg" />
      <Avatar alt="Cindy Baker" src="/static/images/avatar/3.jpg" />
      <Avatar alt="Agnes Walker" src="/static/images/avatar/4.jpg" />
      <Avatar alt="Trevor Henderson" src="/static/images/avatar/5.jpg" />
    </AvatarGroup>
  );
}
```

Total avatars

If you need to control the total number of avatars not shown, you can use the `total` prop.



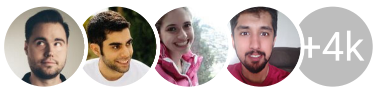
Edit in Chat JS TS Collapse code

```
import Avatar from '@mui/material/Avatar';
import AvatarGroup from '@mui/material/AvatarGroup';

export default function TotalAvatars() {
  return (
    <AvatarGroup total={24}>
      <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
      <Avatar alt="Travis Howard" src="/static/images/avatar/2.jpg" />
      <Avatar alt="Agnes Walker" src="/static/images/avatar/4.jpg" />
      <Avatar alt="Trevor Henderson" src="/static/images/avatar/5.jpg" />
    </AvatarGroup>
  );
}
```

Custom surplus

Set the `renderSurplus` prop as a callback to customize the surplus avatar. The callback will receive the surplus number as an argument based on the children and the `max` prop, and should return a `React.ReactNode`.

The `renderSurplus` prop is useful when you need to render the surplus based on the data sent from the server.

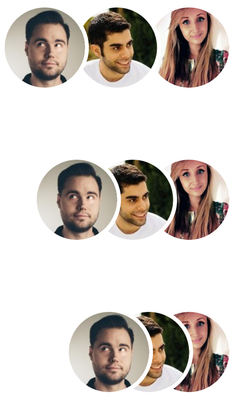


Edit in Chat JS TS Collapse code

```
import Avatar from '@mui/material/Avatar';
import AvatarGroup from '@mui/material/AvatarGroup';

export default function CustomSurplusAvatars() {
  return (
    <AvatarGroup
      renderSurplus={(surplus) => <span>+{surplus.toString()[0]}k</span>}
      total={4251}
    >
      <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
      <Avatar alt="Travis Howard" src="/static/images/avatar/2.jpg" />
      <Avatar alt="Agnes Walker" src="/static/images/avatar/4.jpg" />
      <Avatar alt="Trevor Henderson" src="/static/images/avatar/5.jpg" />
    </AvatarGroup>
  );
}
```

Spacing

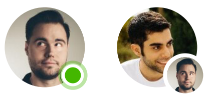
You can change the spacing between avatars using the `spacing` prop. You can use one of the presets (`"medium"`, the default, or `"small"`) or set a custom numeric value.



Edit in Chat JS TS Collapse code

```
import Avatar from '@mui/material/Avatar';
import AvatarGroup from '@mui/material/AvatarGroup';
import Stack from '@mui/material/Stack';

export default function Spacing() {
  return (
    <Stack spacing={4}>
      <AvatarGroup spacing="medium">
        <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
        <Avatar alt="Travis Howard" src="/static/images/avatar/2.jpg" />
        <Avatar alt="Cindy Baker" src="/static/images/avatar/3.jpg" />
      </AvatarGroup>
      <AvatarGroup spacing="small">
        <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
        <Avatar alt="Travis Howard" src="/static/images/avatar/2.jpg" />
        <Avatar alt="Cindy Baker" src="/static/images/avatar/3.jpg" />
      </AvatarGroup>
      <AvatarGroup spacing={24}>
        <Avatar alt="Remy Sharp" src="/static/images/avatar/1.jpg" />
        <Avatar alt="Travis Howard" src="/static/images/avatar/2.jpg" />
        <Avatar alt="Cindy Baker" src="/static/images/avatar/3.jpg" />
      </AvatarGroup>
    </Stack>
  );
}
```

With badge



Edit in Chat JS TS Collapse code

```
import { styled } from '@mui/material/styles';
import Badge from '@mui/material/Badge';
import Avatar from '@mui/material/Avatar';
import Stack from '@mui/material/Stack';

const StyledBadge = styled(Badge)(({ theme }) => ({
  '& .MuiBadge-badge': {
    backgroundColor: '#44b700',
    color: '#44b700',
    boxShadow: `0 0 0 2px ${theme.palette.background.paper}`,
    '&::after': {
      position: 'absolute',
      top: 0,
      left: 0,
      width: '100%',
      height: '100%',
      borderRadius: '50%',
      animation: 'ripple 1.2s infinite ease-in-out',
      border: '1px solid currentColor',
      content: '""',
    },
  },
  '@keyframes ripple': {
    '0%': {
      transform: 'scale(.8)',
      opacity: 1,
    },
    '100%': {
      transform: 'scale(2.4)',
      opacity: 0,
    },
  },
}));

const SmallAvatar = styled(Avatar)(({ theme }) => ({
  width: 22,
```

Avatar upload

Edit in Chat JS TS Hide code

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

export default function UploadAvatars() {
  const [avatarSrc, setAvatarSrc] = React.useState<string | undefined>(undefined);

  const handleAvatarChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    const file = event.target.files?.[0];
    if (file) {
      // Read the file as a data URL
      const reader = new FileReader();
      reader.onload = () => {
        setAvatarSrc(reader.result as string);
      };
      reader.readAsDataURL(file);
    }
  };

  return (
    <ButtonBase
      component="label"
      role={undefined}
      tabIndex={-1} // prevent label from tab focus
      aria-label="Avatar image"
      sx={{
        borderRadius: '40px',
        '&:has(:focus-visible)': {
          outline: '2px solid',
          outlineOffset: '2px',
        },
      }}
    >
      <Avatar alt="Upload new avatar" src={avatarSrc} />
      <input
        type="file"
        accept="image/*"
```

API

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

- `<Avatar />`
- `<AvatarGroup />`
- `<Badge />`

Edit this page

Was this page helpful?