Breakpoints

API that enables the use of breakpoints in a wide variety of contexts.

For optimal user experience, material design interfaces need to be able to adapt their layout at various breakpoints. MUI uses a **simplified** implementation of the original specification.

The breakpoints are used internally in various components to make them responsive, but you can also take advantage of them for controlling the layout of your application through the Grid component.

Default breakpoints

Each breakpoint (a key) matches with a *fixed* screen width (a value):

- **xs,** extra-small: 0px
- **sm,** small: 600px
- **md,** medium: 900px
- **lg,** large: 1200px
- **xl,** extra-large: 1536px

These values can be customized.

CSS Media Queries

CSS media queries are the idiomatic approach to make your UI responsive. The theme provides five styles helpers to do so:

- theme.breakpoints.up(key)
- theme.breakpoints.down(key)
- theme.breakpoints.only(key)
- theme.breakpoints.not(key)
- theme.breakpoints.between(start, end)

In the following demo, we change the background color (red, blue & green) based on the screen width.

```
const styles = (theme) => ({
  root: {
    padding: theme.spacing(1),
    [theme.breakpoints.down('md')]: {
      backgroundColor: theme.palette.secondary.main,
    },
    [theme.breakpoints.up('md')]: {
      backgroundColor: theme.palette.primary.main,
    },
    [theme.breakpoints.up('lg')]: {
      backgroundColor: green[500],
    },
  },
});
```

{{"demo": "MediaQuery.js"}}

JavaScript Media Queries

Sometimes, using CSS isn't enough. You might want to change the React rendering tree based on the breakpoint value, in JavaScript.

useMediaQuery hook

You can learn more on the useMediaQuery page.

Custom breakpoints

You define your project's breakpoints in the `theme.breakpoints` section of your theme.

- `theme.breakpoints.values` : Default to the above values. The keys are your screen names, and the values are the min-width where that breakpoint should start.
- `theme.breakpoints.unit` : Default to `'px'` . The unit used for the breakpoint's values.
- `theme.breakpoints.step` : Default to `5` . The increment divided by 100 used to implement exclusive breakpoints. For example, `{ step: 5 }` means that `down(500)` will result in `'(max-width: 499.95px)'` .

If you change the default breakpoints's values, you need to provide them all:

```
const theme = createTheme({
  breakpoints: {
    values: {
      xs: 0,
      sm: 600,
      md: 900,
      lg: 1200,
      xl: 1536,
    },
  },
});
```

Feel free to have as few or as many breakpoints as you want, naming them in whatever way you'd prefer for your project.

```
const theme = createTheme({
  breakpoints: {
    values: {
      mobile: 0,
      tablet: 640,
      laptop: 1024,
      desktop: 1200,
    },
  },
});
```

If you are using TypeScript, you would also need to use module augmentation for the theme to accept the above values.

```
declare module '@mui/material/styles' {
  interface BreakpointOverrides {
    xs: false; // removes the `xs` breakpoint
    sm: false;
    md: false;
    lg: false;
    xl: false;
    mobile: true; // adds the `mobile` breakpoint
    tablet: true;
    laptop: true;
    desktop: true;
  }
}
```

API

`theme.breakpoints.up(key) => media query`

Arguments

1. `key` (*string* | *number*): A breakpoint key (`xs` , `sm` , etc.) or a screen width number in px.

Returns

`media query` : A media query string ready to be used with most styling solutions, which matches screen widths greater than the screen size given by the breakpoint key (inclusive).

Examples

```
const styles = (theme) => ({
  root: {
    backgroundColor: 'blue',
    // Match [md, ∞)
    //       [900px, ∞)
    [theme.breakpoints.up('md')]: {
      backgroundColor: 'red',
    },
  },
});
```

`theme.breakpoints.down(key) => media query`

Arguments

1. `key` (*string* | *number*): A breakpoint key (`xs` , `sm` , etc.) or a screen width number in px.

Returns

`media query` : A media query string ready to be used with most styling solutions, which matches screen widths less than the screen size given by the breakpoint key (exclusive).

Examples

```
const styles = (theme) => ({
  root: {
    backgroundColor: 'blue',
    // Match [0, md)
    //       [0, 900px)
    [theme.breakpoints.down('md')]: {
      backgroundColor: 'red',
    },
  },
});
```

`theme.breakpoints.only(key) => media query`

Arguments

1. `key` (*string*): A breakpoint key (`xs` , `sm` , etc.).

Returns

`media query` : A media query string ready to be used with most styling solutions, which matches screen widths starting from the screen size given by the breakpoint key (inclusive) and stopping at the screen size given by the next breakpoint key (exclusive).

Examples

```
const styles = (theme) => ({
  root: {
    backgroundColor: 'blue',
    // Match [md, md + 1)
    //       [md, lg)
    //       [900px, 1200px)
    [theme.breakpoints.only('md')]: {
      backgroundColor: 'red',
    },
  },
});
```

`theme.breakpoints.not(key) => media query`

Arguments

1. `key` (*string*): A breakpoint key (`xs` , `sm` , etc.).

Returns

`media query` : A media query string ready to be used with most styling solutions, which matches screen widths stopping at the screen size given by the breakpoint key (exclusive) and starting at the screen size given by the next breakpoint key (inclusive).

Examples

```
const styles = (theme) => ({
  root: {
    backgroundColor: 'blue',
    // Match [xs, md) and [md + 1, ∞)
    //       [xs, md) and [lg, ∞)
    //       [0px, 900px) and [1200px, ∞)
    [theme.breakpoints.not('md')]: {
      backgroundColor: 'red',
    },
  },
});
```

`theme.breakpoints.between(start, end) => media query`

Arguments

1. `start` (*string*): A breakpoint key (`xs` , `sm` , etc.) or a screen width number in px.
2. `end` (*string*): A breakpoint key (`xs` , `sm` , etc.) or a screen width number in px.

Returns

`media query` : A media query string ready to be used with most styling solutions, which matches screen widths greater than the screen size given by the breakpoint key in the first argument (inclusive) and less than the screen size given by the breakpoint key in the second argument (exclusive).

Examples

```
const styles = (theme) => ({
  root: {
    backgroundColor: 'blue',
    // Match [sm, md)
    //       [600px, 900px)
    [theme.breakpoints.between('sm', 'md')]: {
      backgroundColor: 'red',
    },
  },
});
```

Default values

You can explore the default values of the breakpoints using the theme explorer or by opening the dev tools console on this page (`window.theme.breakpoints`).