# Right-to-left

Right-to-left languages such as Arabic, Persian, or Hebrew are supported. To change the direction of MUI components you must follow the following steps.

## Steps

### 1. HTML

Make sure the `dir` attribute is set on the body, otherwise native components will break:

```
<body dir="rtl"></body>
```

As an alternative to the above, you can also wrap your application in an element with the `dir` attribute:

```
function App() {
  return (
    <div dir="rtl">
      <MyComponent />
    </div>
  );
}
```

This can be helpful for creating components to toggle language settings in the live application.

### 2. Theme

Set the direction in your custom theme:

```
import { createTheme } from '@mui/material/styles';

const theme = createTheme({
  direction: 'rtl',
});
```

### 3. Install the rtl plugin

When using either `emotion` or `styled-components`, you need `stylis-plugin-rtl` to flip the styles.

```
npm install stylis stylis-plugin-rtl
```

> **Note**: Only `emotion` is compatible with version 2 of the plugin. `styled-components` requires version 1. If you are using `styled-components` as a styled engine, make sure to install the correct version.

In case you are using `jss` (up to v4) or with the legacy `@mui/styles` package, you need `jss-rtl` to flip the styles.

```
npm install jss-rtl
```

Having installed the plugin in your project, MUI components still require it to be loaded by the style engine instance that you use. Find bellow guides on how you can load it.

## 4. Load the rtl plugin

**4.1 emotion** If you use emotion as your style engine, you should create a new cache instance that uses the `stylis-plugin-rtl` (the default `prefixer` plugin must also be included in order to retain vendor prefixing) and provide that on the top of your application tree. The CacheProvider component enables this:

```
import rtlPlugin from 'stylis-plugin-rtl';
import { CacheProvider } from '@emotion/react';
import createCache from '@emotion/cache';
import { prefixer } from 'stylis';

// Create rtl cache
const cacheRtl = createCache({
  key: 'muirtl',
  stylisPlugins: [prefixer, rtlPlugin],
});

function RTL(props) {
  return <CacheProvider value={cacheRtl}>{props.children}</CacheProvider>;
}
```

**4.2 styled-components** If you use `styled-components` as your style engine, you can use the StyleSheetManager and provide the stylis-plugin-rtl as an item in the `stylisPlugins` property:

```
import { StyleSheetManager } from 'styled-components';
import rtlPlugin from 'stylis-plugin-rtl';

function RTL(props) {
  return (
    <StyleSheetManager stylisPlugins={[rtlPlugin]}>
      {props.children}
    </StyleSheetManager>
  );
}
```

**4.3 JSS** After installing the plugin in your project, you need to configure the JSS instance to load it. The next step is to make the new JSS instance available to all the components in the component tree. The `StylesProvider` component enables this:

```
import { create } from 'jss';
import rtl from 'jss-rtl';
import { StylesProvider, jssPreset } from '@mui/styles';

// Configure JSS
const jss = create({
  plugins: [...jssPreset().plugins, rtl()],
});

function RTL(props) {
  return <StylesProvider jss={jss}>{props.children}</StylesProvider>;
}
```

For more information on the plugin, head to the plugin README. **Note**: Internally, withStyles is using this JSS plugin when `direction: 'rtl'` is set on the theme.

## Demo

*Use the direction toggle button on the top right corner to flip the whole documentation*

{{“demo”: “Direction.js”}}

## Opting out of rtl transformation

### emotion & styled-components

You have to use the template literal syntax and add the `/* @noflip */` directive before the rule or property for which you want to disable right-to-left styles.

```
const AffectedText = styled('div')`
  text-align: left;
`;

const UnaffectedText = styled('div')`
  /* @noflip */
  text-align: left;
`;
```

{{“demo”: “RtlOptOutStylis.js”, “hideToolbar”: true}}

### JSS

If you want to prevent a specific rule-set from being affected by the `rtl` transformation you can add `flip: false` at the beginning.

```
const useStyles = makeStyles(
  (theme) => ({
    affected: {
      textAlign: 'right',
    },
    unaffected: {
      flip: false,
      textAlign: 'right',
    },
  }),
  { defaultTheme },
);
```