类名生成器

在构建时配置类名生成

这个 API 在 `@mui/material` (v5.0.5) 中引入，作为已弃用 `createGenerateClassName` 的替代品

> ⚠️ ***注意**: 这个 API 处于不稳定阶段，将来可能发生变化。*

全局类名前缀

默认情况下， MUI 为每个组件插槽生成一个全局类名。 例如:

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

function App() {
  return <Button>Button</Button>;
}
```

生成以下 HTML：

```
<button
  class="MuiButton-root MuiButton-text MuiButton-textPrimary MuiButton-sizeMedium
MuiButton-textSizeMedium MuiButtonBase-root css-1ujsas3"
>
  Button
</button>
```

使用 `ClassNameGenerator.configure(callback)` 回调函数，可为 MUI 所有生成组件类名添加前缀。

```
+import { outlinedInputClasses } from '@mui/material/OutlinedInput';

   const theme = createTheme({
     components: {
       MuiOutlinedInput: {
         styleOverrides: {
           root: {
-            '& .MuiOutlinedInput-notchedOutline': {
+            // the result will contain the prefix.
+            [`& .${outlinedInputClasses.notchedOutline}`]: {
               borderWidth: 1,
             }
           }
         }
       }
     }
   });
```

返回 HTML 结果

```
<button
  class="foo-bar-MuiButton-root foo-bar-MuiButton-text foo-bar-MuiButton-textPrimary
foo-bar-MuiButton-sizeMedium foo-bar-MuiButton-textSizeMedium foo-bar-MuiButtonBase-
root css-1ujsas3"
>
  Button
</button>
```

组件更名

每个 MUI 组件都有 `${componentName}-${slot}` 类名称格式。 例如， `Chip` 的组件名称为 `MuiChip`，它被用作每一个 `<Chip />` 元素的全局类名称。 您可以删除/更改 `Mui` 前缀，如下所示：

```
import { unstable_ClassNameGenerator } from '@mui/material/utils';

// 在应用程序的根调用这个函数
unstable_ClassNameGenerator.configure((componentName) =>
  componentName.replace('Mui', ''),
);

function App() {
  return <Button>Button</Button>;
}
```

现在， `Mui` 类已经结束。

```
<div
  class="Chip-root Chip-filled Chip-sizeMedium Chip-colorDefault Chip-filledDefault
css-mttbc0"
>
  Chip
</div>
```

> ***注：** 状态类**不是**组件名称，因此不能更改/删除*

注意事项

- `ClassNameGenerator.configure` must be called before any MUI components import.
- 您应该始终使用 `[component]类` 进行主题化/自定义以获得正确的生成类名称。

```
+import { outlinedInputClasses } from '@mui/material/OutlinedInput';

 const theme = createTheme({
   components: {
     MuiOutlinedInput: {
       styleOverrides: {
         root: {
-          '& .MuiOutlinedInput-notchedOutline': {
```

```
+           // the result will contain the prefix.
+           [`& .${outlinedInputClasses.notchedOutline}`]: {
              borderWidth: 1,
            }
          }
        }
      }
    }
  });
```

- 此 API 只能在构建时使用。
- 该配置应用于整个程序的所有组件。 您不能针对应用程序的特定部分。

框架示例

在某些情况下，您可能需要在 `ClassNameGenerator` 导入顶部添加 `/* eslint-disable import/first*/` 来解决报错

```
/* eslint-disable import/first */
import { unstable_ClassNameGenerator as ClassNameGenerator } from
'@mui/material/utils';
```

Next.js

在 `/pages/_app.js` 中使用 ClassNameGenerator。

```
+import { unstable_ClassNameGenerator as ClassNameGenerator } from
'@mui/material/utils';

+ClassNameGenerator.configure((componentName) => componentName.replace('Mui', ''));

 import * as React from 'react';
 import PropTypes from 'prop-types';
 import Head from 'next/head';

 export default function MyApp(props) {
   const { Component, pageProps } = props;

   return (
     <Component {...pageProps} />
   );
 }
```

创建 React 应用

在 `/src/index.js` 中使用 ClassNameGenerator

```
+import { unstable_ClassNameGenerator as ClassNameGenerator } from
'@mui/material/utils';
```

```
+ClassNameGenerator.configure((componentName) => componentName.replace('Mui', ''));

 import * as React from 'react';
 import ReactDOM from 'react-dom';
 import App from './App';

 ReactDOM.render(<App />);
```

Gatsby

在根文件夹中的 `gatsby-ssr.js` 文件内使用 GlassNameGenerator

```
+import { unstable_ClassNameGenerator as ClassNameGenerator } from
"@mui/material/utils";

+ClassNameGenerator.configure((componentName) => componentName.replace('Mui', ''));

 export const wrapPageElement = ({ element }) => {
   return element;
 };
```