API

Uma função que retorna uma função geradora de nome de classe.

`createGenerateClassName([options]) => class name generator`

Uma função que retorna uma função geradora de nome de classe.

Argumentos

1. `options` (*object* [optional]):
    - `options.disableGlobal` (*bool* [opcional]): Padrão `false`. Desabilita a geração de nomes de classes determinísticas.
    - `options.productionPrefix` (*string* [opcional]): Padrão `'jss'`. A string usada para prefixar os nomes de classes em produção.
    - `options.seed` (*string* [opcional]): Padrão `''`. A string usada unicamente para identificar o gerador. Ela pode ser usada para evitar colisões de nomes de classes ao usar vários geradores no mesmo documento.

Retornos

`class name generator`: O gerador que deve ser fornecido ao JSS.

Exemplos

```
import * as React from 'react';
import { StylesProvider, createGenerateClassName } from '@material-ui/styles';

const generateClassName = createGenerateClassName({
  productionPrefix: 'c',
});

export default function App() {
  return <StylesProvider generateClassName={generateClassName}>...</StylesProvider>;
}
```

`createStyles(styles) => styles`

Esta função realmente não "faz nada" em tempo de execução, é apenas uma função de identidade. Sua única finalidade é lidar com a ampliação de tipos do **TypeScript**, ao fornecer regras de estilo para `makeStyles` / `withStyles` que são uma função do `tema`.

Argumentos

1. `styles` (*object*): A styles object.

Retornos

`styles` : Um objeto de estilos.

Exemplos

```
import { createStyles, makeStyles } from '@material-ui/styles';
import { createTheme, ThemeProvider } from '@material-ui/core/styles';

const useStyles = makeStyles((theme: Theme) => createStyles({
  root: {
    backgroundColor: theme.palette.red,
  },
}));

const theme = createTheme();

export default function MyComponent {
  const classes = useStyles();
  return <ThemeProvider theme={theme}><div className={classes.root} />
</ThemeProvider>;
}
```

`makeStyles(styles, [options]) => hook`

Vincula uma folha de estilo a um componente de função usando o padrão **hook**.

Argumentos

1. `styles` (*Function* | *Object*): Uma função que gera os estilos ou um objeto de estilos. Ela será vinculada ao componente. Use a assinatura da função se você precisar ter acesso ao tema. É fornecido como o primeiro argumento.
2. `options` (*object* [optional]):

- `options.defaultTheme` (*object* [opcional]): O tema padrão a ser usado se um tema não for fornecido por meio de um provedor de temas.
- `options.name` (*string* [opcional]): O nome da folha de estilo. Útil para depuração.
- `options.flip` (*bool* [optional]): When set to `false`, this sheet will opt-out the `rtl` transformation. Quando definido para `true`, os estilos são invertidos. Quando definido para `null`, segue `theme.direction`.
- As outras chaves são encaminhadas para o argumento de opções do jss.createStyleSheet ([styles], [options]).

Retornos

`hook` : Um hook. Este hook pode ser usado em uma função que retorna o componente. A documentação geralmente chama esse hook retornado de `useStyles`. Aceita um argumento: as propriedades que serão usadas para "interpolação" na folha de estilo.

Exemplos

```
import * as React from 'react';
import { makeStyles } from '@material-ui/styles';

const useStyles = makeStyles({
  root: {
    backgroundColor: 'red',
    color: (props) => props.color,
  },
});

export default function MyComponent(props) {
  const classes = useStyles(props);
  return <div className={classes.root} />;
}
```

`ServerStyleSheets`

Esta é uma classe auxiliar para manipular a renderização do lado do servidor. Você pode seguir este guia para uma abordagem prática.

```
import ReactDOMServer from 'react-dom/server';
import { ServerStyleSheets } from '@material-ui/styles';

const sheets = new ServerStyleSheets();
const html = ReactDOMServer.renderToString(sheets.collect(<App />));
const cssString = sheets.toString();

const response = `
<!DOCTYPE html>
<html>
  <head>
    <style id="jss-server-side">${cssString}</style>
  </head>
  <body>${html}</body>
</html>
`;
```

`new ServerStyleSheets([options])`

A instanciação aceita um objeto de opções como primeiro argumento.

1. `options` (*object* [optional]): The options are spread as props to the `StylesProvider` component.

`sheets.collect(node) => Elemento React`

O método envolve seu nó React em um elemento provider. Ele coleta as folhas de estilo durante a renderização para que elas possam ser enviadas posteriormente ao cliente.

`sheets.toString() => CSS string`

O método retorna os estilos coletados.

⚠️ Você deve chamar `.collect()` antes de usar este método.

`sheets.getStyleElement() => CSS do elemento React`

O método é uma alternativa para `.toString()` quando você esta renderizando a página inteira com React.

Vincula uma folha de estilos, com uma função de componente, usando o padrão de **componentes estilizados (styled components)**.

`styled(Component)(styles, [options]) => Component`

Vincula uma folha de estilos, com uma função de componente, usando o padrão de **componentes estilizados (styled components)**.

Argumentos

1. `Component` : O componente que será manipulado.
2. `styles` (*Function* | *Object*): Uma função que gera os estilos ou um objeto de estilos. Ela será vinculada ao componente. Use a assinatura da função se você precisar ter acesso ao tema. É fornecido como propriedade do primeiro argumento.
3. `options` (*object* [optional]):

- `options.defaultTheme` (*object* [opcional]): O tema padrão a ser usado se um tema não for fornecido por meio de um provedor de temas.
- `options.withTheme` (*bool* [opcional]): Padrão `false` . Fornecer o objeto `theme` para o componente como uma propriedade.
- `options.name` (*string* [opcional]): O nome da folha de estilo. Useful for debugging. Se o valor não for fornecido, ele tentará usar o nome do componente.
- `options.flip` (*bool* [optional]): When set to `false` , this sheet will opt-out the `rtl` transformation. Quando definido para `true` , os estilos são invertidos. Quando definido para `null` , segue `theme.direction` .
- As outras chaves são encaminhadas para o argumento de opções do jss.createStyleSheet ([styles], [options]).

Retornos

`Component` : The new component created.

Exemplos

```
import * as React from 'react';
import { styled, ThemeProvider } from '@material-ui/styles';
import { createTheme } from '@material-ui/core/styles';

const MyComponent = styled('div')({
  backgroundColor: 'red',
});

const MyThemeComponent = styled('div')(({ theme }) => ({
  padding: theme.spacing(1),
}));
```

```
const theme = createTheme();

export default function StyledComponents() {
  return (
    <ThemeProvider theme={theme}>
      <MyThemeComponent>
        <MyComponent />
      </MyThemeComponent>
    <ThemeProvider>
  );
}
```

`StylesProvider`

Este componente permite que você altere o comportamento da solução de estilo. Ele torna as opções disponíveis na árvore React graças ao contexto.

It should preferably be used at **the root of your component tree**.

Propriedades

Nome	Tipo	Padrão	Descrição
children *	node		Sua árvore de componentes.
disableGeneration	bool	false	Você pode desabilitar a geração dos estilos com esta opção. Pode ser útil ao percorrer a árvore React fora da etapa de renderização de HTML no servidor. Digamos que você esteja usando react-apollo para extrair todas as consultas feitas pela interface do lado do servidor. Você pode acelerar significativamente a varredura com essa propriedade.
generateClassName	func		Gerador de nome de classes do JSS.
injectFirst	bool	false	Por padrão, os estilos são injetados por último no elemento `<head>` da página. Como resultado, eles ganham mais especificidade do que qualquer outra folha de estilo. Se você quiser sobrescrever estilos do Material-UI, defina esta propriedade.
jss	object		Instância do JSS.

Exemplos

```
import * as React from 'react';
import ReactDOM from 'react-dom';
import { StylesProvider } from '@material-ui/styles';

function App() {
  return <StylesProvider jss={jss}>...</StylesProvider>;
} ReactDOM.render(<App />, document.querySelector('#app'));
```

`ThemeProvider`

Este componente tem uma propriedade `theme`, e se torna disponível pela árvore React graças ao contexto. Deve preferencialmente ser usado na **raiz da sua árvore de componentes**.

Propriedades

Nome	Tipo	Padrão	Descrição
children *	node		Sua árvore de componentes.
theme *	union: object | func		Um objeto de tema. Você pode utilizar uma função para receber o tema externo.

Exemplos

```
import * as React from 'react';
import ReactDOM from 'react-dom';
import { ThemeProvider } from '@material-ui/core/styles';

const theme = {};

function App() {
  return <ThemeProvider theme={theme}>...</ThemeProvider>;
} ReactDOM.render(<App />, document.querySelector('#app'));
```

`useTheme() => theme`

Este hook retorna o objeto `theme`, para que possa ser usado dentro de um componente retornado por função.

Retornos

`theme`: O objeto de tema previamente injetado no contexto.

Exemplos

```
import * as React from 'react';
import { useTheme } from '@material-ui/core/styles';

export default function MyComponent() {
  const theme = useTheme();

  return <div>{`spacing ${theme.spacing}`}</div>;
}
```

`withStyles(styles, [options]) => higher-order component`

Vincula uma folha de estilos com um componente usando o padrão de **higher-order component**. Ele não modifica o componente passado para ele; em vez disso, ele retorna um novo componente, com a propriedade `classes`.

Este objeto `classes` contém o nome das classes inseridas no DOM.

Alguns detalhes de implementação que podem ser interessantes para estar ciente:

- Adiciona uma propriedade `classes`, assim você pode substituir, a partir do exterior, os nomes de classe previamente injectados.
- Ela encaminha refs para o componente interno.
- Ele **não** faz copia sobre estáticos. For instance, it can be used to define a `getInitialProps()` static method (next.js).

Argumentos

1. `styles` (*Function* | *Object*): Uma função que gera os estilos ou um objeto de estilos. Ela será vinculada ao componente. Use a assinatura da função se você precisar ter acesso ao tema. É fornecido como o primeiro argumento.
2. `options` (*object* [optional]):

- `options.defaultTheme` (*object* [opcional]): O tema padrão a ser usado se um tema não for fornecido por meio de um provedor de temas.
- `options.withTheme` (*bool* [opcional]): Padrão `false`. Fornecer o objeto `theme` para o componente como uma propriedade.
- `options.name` (*string* [opcional]): O nome da folha de estilo. Useful for debugging. Se o valor não for fornecido, ele tentará usar o nome do componente.
- `options.flip` (*bool* [optional]): When set to `false`, this sheet will opt-out the `rtl` transformation. Quando definido para `true`, os estilos são invertidos. Quando definido para `null`, segue `theme.direction`.
- As outras chaves são encaminhadas para o argumento de opções do jss.createStyleSheet ([styles], [options]).

Retornos

`higher-order component`: Deve ser usado para encapsular o componente.

Exemplos

```
import * as React from 'react';
import { withStyles } from '@material-ui/styles';

const styles = {
  root: {
    backgroundColor: 'red',
  },
};

function MyComponent(props) {
  return <div className={props.classes.root} />;
}

export default withStyles(styles)(MyComponent);
```

Além disso, você pode usar como decoradores dessa forma:

```
import * as React from 'react';
import { withStyles } from '@material-ui/styles';

const styles = {
  root: {
    backgroundColor: 'red',
  },
};

@withStyles(styles)
class MyComponent extends React. Component {
  render() {
    return <div className={this.props.classes.root} />;
  }
}

export default MyComponent;
```

`withTheme(Component) => Component`

Este hook retorna o objeto `theme`, para que possa ser usado dentro de um componente de função.

Argumentos

1. `Component`: O componente que será manipulado.

Retornos

`Component`: O novo componente criado. Encaminha refs para o componente interno.

Exemplos

```
import * as React from 'react';
import { withTheme } from '@material-ui/core/styles';

function MyComponent(props) {
  return <div>{props.theme.direction}</div>;
}

export default withTheme(MyComponent);
```