





useMediaQuery

Este é um hook de consulta de mídia CSS para React. Ele ouve correspondências para uma consulta de mídia no CSS. Permite a renderização de componentes com base no fato de a consulta corresponder ou não.

Algumas das principais características:

-  Tem uma API React idiomática.
-  É performático, ele observa o documento para detectar quando suas consultas de mídia mudam, em vez de pesquisar os valores periodicamente.
-  [1 kB gzipped](#).
-  Suporta a renderização do lado do servidor.

[A paleta](#) com funções de estilo.

Consulta de mídia simples

Você deve fornecer uma consulta de mídia ao primeiro argumento do hook. A string de consulta de mídia pode ser qualquer consulta de mídia CSS válida, por exemplo `'(prefers-color-scheme: dark)'`.

```
{{"demo": "SimpleMediaQuery.js", "defaultCodeOpen": true}}
```

⚠️ Você não pode usar `'print'` devido a limitação de navegadores, por exemplo, este bug presente no [Firefox](#).

Usando auxiliares de ponto de quebra do Material-UI

Você pode usar os [auxiliares de ponto de quebra](#) do Material-UI da seguinte maneira:

```
import { useTheme } from '@material-ui/core/styles';
import useMediaQuery from '@material-ui/core/useMediaQuery';

function MyComponent() {
  const theme = useTheme();
  const matches = useMediaQuery(theme.breakpoints.up('sm'));

  return <span>`theme.breakpoints.up('sm') matches: ${matches}`</span>;
}
```

```
{{"demo": "ThemeHelper.js", "defaultCodeOpen": false}}
```

Como alternativa, você pode usar uma função de retorno, aceitando o tema como um primeiro argumento:

```
import useMediaQuery from '@material-ui/core/useMediaQuery';

function MyComponent() {
  const matches = useMediaQuery((theme) => theme.breakpoints.up('sm'));

  return <span>`theme.breakpoints.up('sm') matches: ${matches}`</span>;
}
```

Você pode usar [json2mq](#) para gerar uma string de consulta de mídia a partir de um objeto JavaScript.

Usando a sintaxe JavaScript

Você pode usar [json2mq](#) para gerar uma string de consulta de mídia a partir de um objeto JavaScript.

```
{ "demo": "JavaScriptMedia.js", "defaultCodeOpen": true }
```

Testando

Você precisa de uma implementação de [matchMedia](#) em seu ambiente de teste.

Por exemplo, [jsdom não suporta ainda](#). Você deve usar um polyfill para isso. É recomendável usar [css-mediaquery](#) para emular.

```
import mediaQuery from 'css-mediaquery';

function createMatchMedia(width) {
  return (query) => ({
    matches: mediaQuery.match(query, { width }),
    addListener: () => {},
    removeListener: () => {},
  });
}

describe('MeusTestes', () => {
  beforeAll(() => {
    window.matchMedia = createMatchMedia(window.innerWidth);
  });
});
```

Renderização somente do lado do cliente

Para executar a hidratação no lado do servidor, o hook precisa renderizar duas vezes. Uma primeira vez com `false`, o valor do servidor e uma segunda vez com o valor resolvido. Este ciclo de renderização de dupla passagem tem uma desvantagem. É mais lento. Você pode definir a opção `noSsr` para `true` se você estiver fazendo renderização **somente no lado cliente**.

```
const matches = useMediaQuery('(min-width:600px)', { noSsr: true });
```

ou pode ativar globalmente com o tema:

```
const theme = createTheme({
  components: {
    MuiUseMediaQuery: {
      defaultProps: {
        noSsr: true,
      },
    },
  },
});
```

Renderização do lado servidor

⚠ *Renderização do lado servidor e consultas de mídia do lado cliente são fundamentalmente conflitantes. Esteja ciente da escolha. O suporte só pode ser parcial.*

Tente confiar em consultas de mídia CSS do lado do cliente primeiro. Por exemplo, você poderia usar:

- `<Box display>`
- `themes.breakpoints.up(x)`
- or `sx prop`

Se nenhuma das alternativas acima for uma opção, você poderá continuar lendo esta seção da documentação.

Primeiro, você precisa adivinhar as características da solicitação do cliente, no servidor. Você tem a opção entre usar:

- **User agent.** Analise a string do user agent do cliente para extrair informações. É recomendável usar [ua-parser-js](#) para analisar o user agent.
- **Client hints.** Leia as dicas que o cliente está enviando para o servidor. Esteja ciente de que esse recurso [não é suportado em qualquer lugar](#).

Por fim, você precisa fornecer uma implementação de [matchMedia](#) para o `useMediaQuery` com as características adivinhadas anteriormente. É recomendável usar [css-mediaquery](#) para emular o `matchMedia`.

Por exemplo, no lado do servidor:

```
import ReactDOMServer from 'react-dom/server';
import parser from 'ua-parser-js';
import mediaQuery from 'css-mediaquery';
import { ThemeProvider } from '@material-ui/core/styles';

function handleRender(req, res) {
  const deviceType = parser(req.headers['user-agent']).device.type || 'desktop';
  const ssrMatchMedia = query => ({
    matches: mediaQuery.match(query, {
      // O CSS estimado pelo navegador.
      width: deviceType === 'mobile' ? '0px' : '1024px',
    }),
  });

  const theme = createTheme({
    components: {
      // Modifica as opções padrão de useMediaQuery
      MuiUseMediaQuery: {
        defaultProps: {
          ssrMatchMedia,
        },
      },
    },
  });

  const html = ReactDOMServer.renderToString(
    <ThemeProvider theme={theme}>
      <App />
    </ThemeProvider>
  );
}
```

```
    </ThemeProvider>,
  );

  // ...
}
```

```
{{"demo": "ServerSide.js", "defaultCodeOpen": false}}
```

Certifique-se de fornecer a mesma implementação de mídia de correspondência customizada para o lado do cliente para garantir uma correspondência de hidratação.

Migrando de `withWidth()`

O componente de ordem superior `withWidth()` injeta a largura da tela da página. Você pode reproduzir o mesmo comportamento com o hook `useWidth`:

```
{{"demo": "UseWidth.js"}}
```

API

`useMediaQuery(query, [options]) => matches`

Argumentos

1. `query` (*string* | *func*): A string representing the media query to handle or a callback function accepting the theme (in the context) that returns a string.
2. `options` (*object* [optional]):
 - `options.defaultMatches` (*bool* [optional]): As `window.matchMedia()` is unavailable on the server, we return a default matches during the first mount. O valor padrão é `false`.
 - `options.matchMedia` (*func* [optional]): You can provide your own implementation of *matchMedia*. Isso pode ser usado para manipular uma janela iframe com conteúdo.
 - `options.noSsr` (*bool* [opcional]): Padrão `false`. Para executar a hidratação no lado do servidor, o hook precisa renderizar duas vezes. Uma primeira vez com `false`, o valor do servidor e uma segunda vez com o valor resolvido. Este ciclo de renderização de dupla passagem tem uma desvantagem. É mais lento. Você pode definir esta opção para `true` se você estiver fazendo renderização **somente no lado cliente**.
 - `options.ssrMatchMedia` (*func* [optional]): You can provide your own implementation of *matchMedia* in a [server-side rendering context](#).

Nota: Você pode alterar as opções padrão usando [default_props](#), este recurso pertence ao tema através da chave `MuiUseMediaQuery`.

Retornos

`matches`: Matches é `true` se o documento coincidir com a consulta de mídia, e `false` quando isso não ocorrer.

Exemplos

```
import * as React from 'react';
import useMediaQuery from '@material-ui/core/useMediaQuery';
```

```
export default function SimpleMediaQuery() {  
  const matches = useMediaQuery('(min-width:600px)');  
  
  return <span>`(min-width:600px) matches: ${matches}`</span>;  
}
```