Implementación de Media Queries en React

Julian F. Latorre

Contents

1 Introducción

Las media queries son una herramienta poderosa en el diseño web responsivo, permitiendo ajustar el estilo y la estructura de una página web basándose en características del dispositivo como el ancho de la pantalla. En React, podemos implementar media queries de varias maneras para crear aplicaciones que se adapten a diferentes tamaños de pantalla.

2 Marco Conceptual

2.1 Diseño Responsivo

El diseño responsivo es una aproximación al diseño web que hace que las páginas web se rendericen bien en una variedad de dispositivos y tamaños de ventana o pantalla. Es crucial para proporcionar una experiencia de usuario óptima en dispositivos que van desde teléfonos móviles hasta pantallas de escritorio de gran tamaño.

2.2 Media Queries

Las media queries son una característica de CSS3 que permite aplicar estilos condicionalmente basándose en varias características del dispositivo, principalmente el ancho, alto y orientación de la ventana del navegador. Son la base técnica que permite implementar diseños responsivos.

2.3 Breakpoints

Los breakpoints son puntos específicos en el espectro de tamaños de pantalla donde el diseño de la interfaz de usuario cambia para acomodarse mejor al nuevo tamaño. Estos puntos de quiebre se definen típicamente en términos de ancho de pantalla y se utilizan en conjunto con las media queries para crear diseños adaptables.

3 Breakpoints Comunes

Aunque los breakpoints pueden variar según las necesidades específicas de cada proyecto, existen algunos puntos de quiebre comúnmente utilizados en el desarrollo web:

- **Mobile**: ¡ 576px
- **Tablet**: 576px - 768px
- **Desktop**: 769px - 992px
- **Large Desktop**: ¿ 992px

Es importante notar que estos son solo puntos de referencia y pueden ajustarse según las necesidades específicas del diseño y la audiencia objetivo.

4 Implementación de Breakpoints en React

En React, podemos implementar breakpoints de varias maneras. A continuación, exploraremos tres métodos principales, cada uno con sus propias ventajas y casos de uso.

5 Preparación del Entorno

Antes de comenzar, asegúrate de tener un proyecto React configurado. Si no lo tienes, puedes crear uno usando Create React App:

```
npx create-react-app media-queries-demo
cd media-queries-demo
npm start
```

6 Método 1: CSS Tradicional

El método más simple es usar media queries directamente en tu CSS.

6.1 Paso 1: Crear un archivo CSS

Crea un archivo CSS, por ejemplo, `styles.css`:

```
.container {
  padding: 20px;
  background-color: #f0f0f0;
}

@media (max-width: 600px) {
  .container {
    padding: 10px;
    background-color: #e0e0e0;
  }
}
```

6.2 Paso 2: Importar y usar el CSS

Importa el CSS en tu componente React:

```
import React from 'react';
import './styles.css';

const ResponsiveComponent = () => {
  return (
    <div className="container">
      <h1>Contenido Responsivo</h1>
```

```
      <p>Este contenido se adapta a diferentes tama os de pantalla
    .</p>
    </div>
  );
};

export default ResponsiveComponent;
```

7 Método 2: CSS-in-JS con Styled Components

Styled Components es una biblioteca popular para CSS-in-JS que facilita el uso de media queries en componentes React.

7.1 Paso 1: Instalar Styled Components

```
npm install styled-components
```

7.2 Paso 2: Crear componentes estilizados con media queries

```
import React from 'react';
import styled from 'styled-components';

const Container = styled.div`
  padding: 20px;
  background-color: #f0f0f0;

  @media (max-width: 600px) {
    padding: 10px;
    background-color: #e0e0e0;
  }
`;

const ResponsiveComponent = () => {
  return (
    <Container>
      <h1>Contenido Responsivo</h1>
      <p>Este contenido se adapta usando Styled Components.</p>
    </Container>
  );
};

export default ResponsiveComponent;
```

8 Método 3: Hook personalizado para Media Queries

Podemos crear un hook personalizado para manejar media queries de forma dinámica.

8.1 Paso 1: Crear el hook useMediaQuery

```
import { useState, useEffect } from 'react';

const useMediaQuery = (query) => {
  const [matches, setMatches] = useState(false);

  useEffect(() => {
    const media = window.matchMedia(query);
    if (media.matches !== matches) {
      setMatches(media.matches);
    }
    const listener = () => setMatches(media.matches);
    media.addListener(listener);
    return () => media.removeListener(listener);
  }, [matches, query]);

  return matches;
};

export default useMediaQuery;
```

8.2 Paso 2: Usar el hook en un componente

```
import React from 'react';
import useMediaQuery from './useMediaQuery';

const ResponsiveComponent = () => {
  const isMobile = useMediaQuery('(max-width: 600px)');

  return (
    <div style={{
      padding: isMobile ? '10px' : '20px',
      backgroundColor: isMobile ? '#e0e0e0' : '#f0f0f0',
    }}>
      <h1>Contenido Responsivo</h1>
      <p>Este contenido se adapta usando un hook personalizado.</p>
    </div>
  );
};

export default ResponsiveComponent;
```